MANAGING AND MINING SENSOR DATA

MANAGING AND MINING SENSOR DATA

Edited by
CHARU C. AGGARWAL
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

Kluwer Academic Publishers
Boston/Dordrecht/London

Contents

Preface

Sensor data has become pervasive in recent years because of the popularization and wider availability of sensor technology through cheaper embedded sensor devices and RFID technology. Sensors produce large volumes of data continuously over time, and this leads to numerous computational challenges. Such challenges arise both from *accuracy* and *scalability* perspectives.

The scalability challenges of sensor data analytics have reached extraordinary proportions, with the increasing proliferation of ubiquitous and embedded sensors and mobile devices, each of which can potentially generate large streams of data. Many of these devices are internet-connected. This has enabled greater possibilities for different kinds of distributed data sharing and analytics. It has been estimated that the number of internet-connected devices has exceeded the number of people on the planet since 2008. Therefore, it is foreseeable, that in the coming years, machine generated data will dominate human-generated data by orders of magnitude, and this gap is only likely to increase with time. In this context, the challenges associated with scalable and real-time management and mining of sensor data are likely to become even more significant with time.

Sensor data mining is a relatively new area, which is now reaching a certain level of maturity. In spite of this, the data analytics researchers have often remained disconnected from the networking issues which arise during data collection and processing. While the focus of this book is clearly on the data analytics side, we have taken special care to emphasize the impact of the network-specific issues on data processing.

This book discusses the key issues in the collection, modeling and processing of sensor data. The content of the book is carefully designed to cover the area of sensor data mining comprehensively. Each chapter is written as a survey by a well known researcher from the field, so as to cover this area comprehensively. Emphasis is also provided on different applications of sensor networks. A number of newer applications such as social sensing and the internet-of-things are also discussed in this book.

The book is intended for graduate students, researchers and professors. Emphasis has been placed on simplifying the material and making it more accessible. The material in the book can be helpful to both beginners on the subject and advanced researchers. At the same time, the latest topics are covered in significant detail. It is hoped that this book will provide a comprehensive overview of the research in this field. It will be a useful guide to students, researchers and practitioners.

Chapter 1

AN INTRODUCTION TO SENSOR DATA ANALYTICS

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
charu@us.ibm.com

Abstract The increasing advances in hardware technology for sensor processing and mobile technology has resulted in greater access and availability of sensor data from a wide variety of applications. For example, the commodity mobile devices contain a wide variety of sensors such as GPS, accelerometers, and other kinds of data. Many other kinds of technology such as RFID-enabled sensors also produce large volumes of data over time. This has lead to a need for principled methods for efficient sensor data processing. This chapter will provide an overview of the challenges of sensor data analytics and the different areas of research in this context. We will also present the organization of the chapters in this book in this context.

Keywords: Sensor data, stream processing

1. Introduction

Recent years have seen tremendous advances in hardware technology such as the development of miniaturized sensors, GPS-enabled devices, pedometers, and accelerometers, which can be used to collect different kinds of data [6]. This has lead to a deluge of tremendous amounts of real-time data, which can be mined for a variety of analytical insights. The costs of sensor hardware has been consistently going down over the past few years. Furthermore, many data collection technologies [5] such as RFID have been enabled in a very cost-effective way, as a result of which the *scale* of the collection process has become enormous. Sensor

data is produced in the context of a wide variety of applications such as the following:

- A wide variety of mobile devices are now GPS-enabled. This has lead to unprecedented opportunities in the context of several applications such as social sensing [4]. GPS data is also available in the context of many location-aware devices and applications.

- The decreasing cost of RFID tags has lead to tremendous volumes of RFID data. The cost of an RFID tag is now in the range of under 5 cents. This has allowed cost-effective deployment of RFID tags on products of even modest price. RFID data poses numerous challenges because of the tremendous amounts of noise in the collected data [5].

- Numerous military applications use a wide variety of sensors in order to track for unusual events or activity. This could include visual or audio cameras, or seismometers for tracking movements of large objects [9].

- Sensors are also deployed in the context of a wide variety of environmental applications, such as detecting weather and climate trends [7], and tracking pollution levels in water networks [11].

Sensor data brings numerous challenges with it in the context of data collection, storage and processing. This is because sensor data processing often requires efficient and real-time processing from massive volumes of possibly uncertain data. Some of these challenges may be enumerated as follows:

- Data collection is a huge challenge in the context of sensor processing because of the natural errors and incompleteness in the collection process. Sensors often have limited battery life, because of which many of the sensors in a network may not be able to collect or transmit their data over large periods of time. The errors in the underlying data may lead to uncertainty of the data representation [8]. Therefore, methods need to be designed to process the data in the presence of uncertainty.

- Sensors are often designed for applications which require *real-time processing*. This requires the design of efficient methods for stream processing [1]. Such algorithms need to be executed in one pass of the data, since it is typically not often possible to store the entire data set because of storage and other constraints.

- The large volumes of data lead to huge challenges in terms of storage and processing of the data. It has been estimated that since 2008, the number of internet-connected devices has exceeded the number of people on the planet. Thus, it is clear that the amount of machine generated data today greatly exceeds the amount of human generated data, and this gap is only likely to increase in the forseeable future. This is widely known as the *big data* problem in the context of analytical applications [10], or the *information overload problem* in stream processing.

- In many cases, it is critical to perform *in-network processing*, wherein the data is processed within the network itself, rather than at a centralized service. This needs effective design of distributed processing algorithms, wherein queries and other mining algorithm can be processed within the network in real time [12].

In this book, we will provide an overview of the key areas of research in sensor processing, as they related to these challenges. We will also study a number of new applications of sensor data such as social sensing, mobile data processing, RFID processing, and the internet of things.

This chapter is organized as follows. In the next section, we will discuss the key areas of research in sensor processing, as they relate to the afore-mentioned challenges. We will also relate the different research areas to these challenges. Section 3 discusses the conclusions and summary.

2. Research in Sensor Processing

The research issues in the area of sensor processing arise along all stages of the pipeline, beginning from data collection, cleaning, data management, and knowledge discovery and mining. Furthermore, many research issues arise in the context of in-network processing, which are specific to the particular application domain. The specificity to the application domain may arise in the context of other parts of the pipeline as well. Therefore, we summarize the key research issues which arise in the context of sensor data processing as follows:

- **Data Collection and Cleaning Issues:** Numerous issues arise in the context of collection of sensor data. Sensor data is inherently noisy and uncertain, and may either have missed readings or redundant readings depending upon the application domain. For example, in the context of RFID data, almost 30% of the readings are dropped, and multiple sensors may track the same RFID object. In the context of battery-driven sensors, numerous errors may

arise during data transmission, and there may also be significant incompleteness because of limited battery life.

- **Data Management Issues:** The large volumes of collected data poses significant challenges for the collected data. Sometimes, the volume of the data is so large, that it may be impractical to store the entire raw data, and it may be desirable to either compress or drop portions of the data. What parts of the data should be dropped or compressed? The errors and uncertainty in sensor data, have spurred the development of algorithms for uncertain database management [2].

- **Sensor Data Mining and Processing:** The large volumes of sensor data necessitate the design of efficient one-pass algorithms which require at most one scan of the data. These are traditionally referred to as data stream mining algorithms. Furthermore, it may sometimes be advantageous to perform *in-network processing*, which can perform partial processing of the data in the network before sending these results on to a higher level of storage.

- **Application-Specific Issues:** Sensor data can arise in many domains such as retail data (RFID), military sensor networks, astronomy, the environment, and mobile data. Different domains may lead to different issues in the context of storage and processing. For example, RFID data may have larger levels of redundancy and uncertainty, whereas mobile data mining applications may require spatio-temporal mining techniques.

The different chapters of this book will study these different aspects of sensor stream processing. Therefore, the book will be organized so as to comprehensively study these different aspects. The different topics covered by the chapters of this book are as follows:

Data Collection and Management Issues The key data collection and management issues are discussed in Chapter 2. This chapter discusses some of the key database management aspects, which have recently been designed in the context of sensor data. Issues involving data uncertainty and query processing are discussed in this chapter, especially in the context of sensor data. The area of indexing and query processing is very important in the context of sensor data, and therefore we have also designed chapters specifically for this topic.

Query Processing of Sensor Data Sensor data poses numerous challenges from the perspective of indexing and query processing, be-

cause of the massive volume of the data which is received over time. A special case of query processing in sensor data is that of event detection, wherein continuous queries are posed on the sensor data in order to detect the underlying events. The main challenge in event processing is that the high level semantic events are often a complex function of the underlying raw sensor data. In some cases, the event-query cannot be posed exactly, since the event detection process is ambiguously related to the underlying data. Methods for query processing of sensor data are discussed in Chapter 3. Specialized methods for event processing of sensor data are discussed in Chapter 4.

Mining Sensor Data A variety of data mining methods such as clustering, classification, frequent pattern mining, and outlier detection are often applied to sensor data in order to extract actionable insights. This data usually needs to be compressed and filtered for more effective mining and analysis. The main challenge is that conventional mining algorithms are often not designed for real time processing of the data. Therefore, new algorithms for sensor data stream processing need to perform the analytics in a single pass in real time. In addition, the sensor scenario may often require *in-network* processing, wherein the data is processed to higher level representations before further processing. This reduces the transmission costs, and the data overload from a storage perspective. The problems of stream compression [3] and stream mining are therefore tightly integrated together from an efficiency perspective. For example, compression and hidden variable modeling provides summarized representations which can be leveraged for applications such as forecasting and outlier analysis. A survey of methods for dimensionality reduction, compression and filtering of sensor streams is provided in Chapter 5. This chapter studies the issue of stream correlation analysis, compression across streams in terms of hidden variables, and compression across time in a given stream. The application of these concepts to a few stream mining problems is also studied in the same chapter. A number of methods for real-time sensor stream mining, processing and analytics are discussed in Chapters 6 and 7. Specific methods for mining sensor streams in the distributed setting are presented in Chapter 8.

Social Sensing Applications and Mobile Data The popularity of mobile phones and other sensor-enabled devices has lead to a plethora of "socially-aware data" which can be mined in the context of a wide variety of applications. This trend has lead to the integration of sensors and dynamic social networks. A number of architectural, privacy and trust issues arise in the collection of socially aware sensor data. These

issues are discussed in detail in Chapter 9. The chapter also discusses the issues of mining the different kinds of GPS- and content-based data generated in such applications.

Much of the data in social sensing applications often contains GPS trajectory data. Mobile data has a number of characteristics, which can be exploited in order to create more efficient methods for clustering, classification, anomaly detection, and pattern mining. Therefore, we have included a chapter which discusses algorithms for mobile data analysis in detail. Chapter 10 provides a detailed discussion of a wide variety of indexing and mining algorithms in the context of mobile data.

RFID Data and the Internet of Things The trend towards ubiquitous and embedded sensing has lead to a natural focus on machine-to-machine (M2M) paradigms in sensor processing. These paradigms use small RFID sensors to collect data about many smart objects. The data generated from such applications can be shared by different devices for heterogeneous fusion and inference, especially if the devices are connected to the internet. A number of issues also arise about how such devices can be effectively discovered and used by different network participants. Chapter 11 provides an overview on RFID applications for collecting such data. Issues about how such data can be used in the context of the internet of things are discussed in Chapter 12.

Software Bug Tracing in Sensor Networks Most of the aforementioned chapters provide application-specific insights on the basis of the collected data. Sensors also produce diagnostic data, which can be used in order to determine diagnostic bugs within the sensor software. Thus, this kind of mining process can be used in order to improve the performance of the underlying sensor network. A survey of methods and algorithms for software bug tracing in sensor networks is provided in Chapter 13.

Healthcare Applications Sensor data has found increasing application in the health care domain. A wide variety of Intensive Care Unit (ICU) applications use sensors such as ECG, EEG, blood pressure monitors, respiratory monitors, and a wide variety of other sensors in order to track the condition of the patient. The volume of such data is extremely large and the inferences from such data need to be performed in a time-critical fashion. Chapter 14 provides an overview of sensor mining applications in the context of health-care data.

Environmental and Climate Applications A wide variety of sensors are used in order to track environmental and sensor data. A tremendous amount of sensor data is available through satellite sensing, and other more conventional forms of sensing. Such data can be used in order to determine the short terms and long terms trends in climate change, and other environmental applications, such as detecting changes in land cover. Chapter 15 provides an overview of how sensor data may be used in the context of environmental and climate applications.

3. Conclusions and Summary

In this chapter, we provided an overview of the challenges and the key areas of research in sensor processing. We also presented the organization of this book, as it relates to these challenges. The ubiquity and volume of sensor data is likely to increase over time, as more and more applications containing sensor data become widely available. A number of emerging areas of research such as social sensing have brought the use of sensor data within the reach of the masses, because of their incorporation in commoditized devices such as mobile phones. Furthermore, newer applications such as the internet of things have lead to a greater focus on the effective storage and processing of sensor data. This book will discuss all of these challenges in a holistic and integrated way.

References

[1] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[2] C. C. Aggarwal. Managing and Mining Uncertain Data, *Springer*, 2009.

[3] C. C. Aggarwal, P. S. Yu. A Survey of Synopsis Construction Algorithms in Data Streams, *Data Streams: Models and Algorthms*, Springer, 2007.

[4] C. C. Aggarwal, T. Abdelzaher. Integrating Sensors and Social Networks, *Social Network Data Analytics, Springer*, 2011.

[5] C. C. Aggarwal, J. Han. A Survey of RFID Data Processing, *Managing and Mining Sensor Data*, Springer, 2013.

[6] I. Akylldiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. A Survey on Sensor Networks, *IEEE Communications Magazine*, 2002.

[7] D. Culler, D. Estrin. M. Srivastava. Guest Editor's Introduction: An Overview of Sensor Networks. *Computer*, 37(8), 2004.

[8] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong. Model-driven data acquisition in sensor networks, *VLDB Conference*, 2004.

[9] D. Hall, J. Llinas. An Introduction to Multi-sensor data fusion, *Proceedings of the IEEE*, 85(1), 1997.

[10] S. Lohr. The age of big data, *New York Time Sunday Review*, February 12, 2012. `http://www.nytimes.com/2012/02/12--/sunday-review/big-datas-impact-in-the-world.html?pagewanted=all`

[11] A. Ostfeld(et.al.) The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 556–568, 2008. [Online]. Available: `http://link.aip.org/link/?QWR/134/556/1`

[12] Y. Yao, J. E. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, Vol. 31(3), September 2002.

Chapter 2

A SURVEY OF MODEL-BASED SENSOR DATA ACQUISITION AND MANAGEMENT

Saket Sathe
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
saket.sathe@epfl.ch

Thanasis G. Papaioannou
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
thanasis.papaioannou@epfl.ch

Hoyoung Jeung
SAP Research
Brisbane, Australia
hoyoung.jeung@sap.com

Karl Aberer
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
karl.aberer@epfl.ch

Abstract In recent years, due to the proliferation of sensor networks, there has been a genuine need of researching techniques for sensor data acquisition and management. To this end, a large number of techniques have emerged that advocate *model-based* sensor data acquisition and management. These techniques use mathematical models for performing various, day-to-day tasks involved in managing sensor data. In this chapter, we survey the state-of-the-art techniques for model-based sensor data acquisition and management. We start by discussing the techniques for

acquiring sensor data. We, then, discuss the application of models in sensor data cleaning; followed by a discussion on model-based methods for querying sensor data. Lastly, we survey model-based methods proposed for data compression and synopsis generation.

Keywords: model-based techniques, data acquisition, query processing, data cleaning, data compression.

1. Introduction

In recent years, there has been tremendous growth in the data generated by sensor networks. Equivalently, there are pertinent techniques proposed in recent literature for efficiently acquiring and managing sensor data. One important category of techniques that have received significant attention are the model-based techniques. These techniques use mathematical models for solving various problems pertaining to sensor data acquisition and management. In this chapter, we survey a large number of state-of-the-art model-based techniques for sensor data acquisition and management. Model-based techniques use various types of models: statistical, signal processing, regression-based, machine learning, probabilistic, or time series. These models serve various purposes in sensor data acquisition and management.

It is well-known that many physical attributes, like, ambient temperature or relative humidity, vary smoothly. As a result of this smoothness, sensor data typically exhibits the following properties: (a) it is continuous (although we only have a finite number of samples), (b) it has finite energy or it is band-limited, (c) it exhibits Markovian behavior or the value at a time instant depends only on the value at a previous time instant. Most model-based techniques exploit these properties for efficiently performing various tasks related to sensor data acquisition and management.

In this chapter, we consider four broad categories of sensor data management tasks: data acquisition, data cleaning, query processing, and data compression. These tasks are pictorially summarized in the toy example shown in Figure 2.1. From Figure 2.1, it is interesting to note how a single type of model (linear) can be used for performing these various tasks. For each task considered in this chapter, we extensively discuss various, well-researched model-based solutions. Following is the detailed discussion on the sensor data management tasks covered in this chapter:

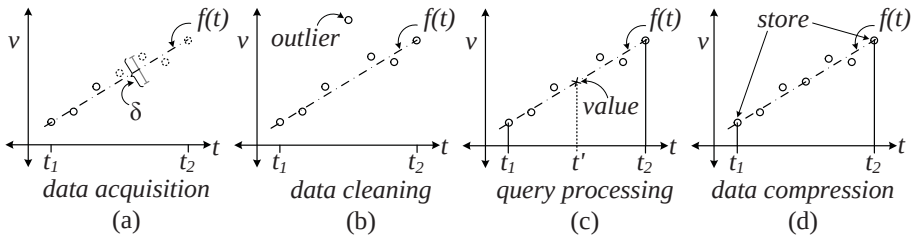


Figure 2.1. Various tasks performed by models-based techniques. (a) to improve acquisitional efficiency, a function is fitted to the first three sensor values, and the remaining values (shown dotted) are not acquired, since they are within a threshold δ, (b) data is cleaned by identifying outliers after fitting a linear model, (c) a query requesting the value at time t' can be answered using interpolation, (d) only the first and the last sensor value can be stored as compressed representation of the sensor values.

- **Data Acquisition:** Sensor data acquisition is the task responsible for efficiently acquiring samples from the sensors in a sensor network. The primary objective of the sensor data acquisition task is to attain energy efficiency. This objective is driven by the fact that most sensors are battery-powered and are located in inaccessible locations (e.g., environmental monitoring sensors are sometimes located at high altitudes and are surrounded by highly inaccessible terrains). In the literature, there are two major types of acquisition approaches: pull-based and push-based. In the pull-based approach, data is only acquired at a user-defined frequency of acquisition. On the other hand, in the push-based approach, the sensors and the base station agree on an expected behavior; sensors only send data to the base station if the sensor values deviate from such expected behavior. In this chapter, we cover a representative collection of model-based sensor data acquisition approaches [2, 12, 17, 16, 18, 27, 28, 41, 66].

- **Data Cleaning:** The data obtained from the sensors is often erroneous. Erroneous sensor values are mainly generated due to the following reasons: (a) intermittent loss of communication with the sensor, (b) sensor's battery is discharged, (c) other types of sensor failures, for example, snow accumulation on the sensor, etc. Model-based approaches for data cleaning often use a model to infer the most probable sensor value. Then the raw sensor value is marked erroneous or outlier if the raw sensor value deviates significantly from the inferred sensor value. Another important approach for data cleaning is known as declarative data cleaning [32, 46, 54].

In this approach, the user registers SQL-like queries that define constraints over the sensor values. Sensor values are marked as outliers when these constraints are violated. In addition to these methods, we also discuss many other data cleaning approaches [31, 73, 23, 21, 52, 65]

- **Query Processing:** Obtaining desired answers, by processing queries is another important aspect in sensor data management. In this chapter, we discuss the most significant model-based techniques for query processing. One of the objectives of these techniques is to process queries by accessing or generating minimal amount of data [64, 5]. Model-based methods that access/generate minimal data, and also handle missing values in data, use models for creating an abstraction layer over the sensor network [18, 33]. Other approaches model the sensor values by a hidden Markov model (HMM), associating state variables to the sensor values. It, then, becomes efficient to process queries over the state variables, which are less in number as compared to the sensor values [5]. Furthermore, there are approaches that use dynamic probabilistic models (DPMs) for modeling spatio-temporal evolution of the sensor data [33, 29]. In these approaches, the estimated DPMs are used for query processing.

- **Data Compression:** It is well-known that large quantity of sensor data is being generated by every hour. Therefore, eliminating redundancy by compressing sensor data for various purposes (like, storage, query processing, etc.) becomes one of the most challenging tasks. Model-based sensor data compression proposes a large number of techniques, mainly from the signal processing literature, for this task [1, 72, 22, 53, 7]. Many approaches assume that the user provides an accuracy bound, and based on this bound the sensor data is approximated, resulting in compressed representations of the data [24]. A large number of other techniques exploit the fact that sensor data is often correlated; thus, this correlation can be used for approximating one data stream with another [24, 67, 49, 3].

This chapter is organized as follows. In Section 2, we define the preliminaries that are assumed in the rest of the chapter, followed by a discussion of important techniques for sensor data acquisition. In Section 3, we survey model-based sensor data cleaning techniques, both on-line and archival. Model-based query processing techniques are discussed in Section 4. In Section 5, model-based compression techniques

are surveyed. At the end, Section 6 contains a summary of the chapter along with conclusions.

2. Model-Based Sensor Data Acquisition

In this section, we discuss various techniques for model-based[1] sensor data acquisition. Particularly, we discuss pull- and push-based sensor data acquisition methods. In general, model-based sensor data acquisition techniques are designed for tackling the following challenges:

Energy Consumption: Obtaining values from a sensor requires high amount of energy. In contrast, since most sensors are battery-powered, they have limited energy resources. Thus, a challenging task is to minimize the number of samples obtained from the sensors. Here, models are used for selecting sensors, such that user queries can be answered with reasonable accuracy using the data acquired from the selected sensors [2, 17, 16, 27, 28].

Communication Cost: Another energy-intensive task is to communicate the sensed values to the base station. There are, therefore, several model-based techniques proposed in the literature for reducing the communication cost, and maintaining the accuracy of the sensed values [41, 18, 66, 12].

Table 2.1. Summary of notations.

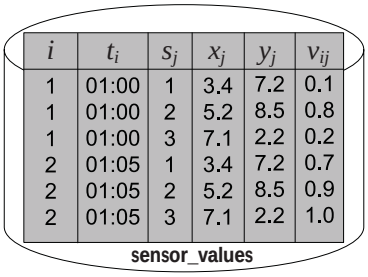
Symbol	Description
$\mathbf{S}$	Sensor network consisting of sensors s_j, where $j = (1, \ldots, m)$.
s_j	Sensor identifier for a sensor in $\mathbf{S}$.
v_{ij}	Sensor value observed by the sensor s_j at time t_i, such that $v_{ij} \in \mathbb{R}$.
v_i	Row vector of all sensor values observed at time t_i, such that $v_i \in \mathbb{R}^m$.
V_{ij}	Random variable associated with the sensor value v_{ij}.

2.1 Preliminaries

We start by describing our model of a sensor network and establishing the notation that is utilized in the rest of the chapter. The sensor network considered in this chapter consists of a set of stationary sensors $\mathbf{S} = \{s_j | 1 \leq j \leq m\}$. The value sensed by a sensor s_j at time t_i is denoted as v_{ij}, which is a real number. In addition, note that we use s_j, where $j = (1, \ldots, m)$, as sensor identifiers. In certain cases the sampling interval could be uniform, that is, $t_{i+1} - t_i$ is same for all the values of

[1] We use *model-based* and *model-driven* interchangeably.

$i \geq 1$. In such cases, the time stamps t_i become irrelevant, and it is sufficient to use only the index i for denoting the time axis.



i	t_i	s_j	x_j	y_j	v_{ij}
1	01:00	1	3.4	7.2	0.1
1	01:00	2	5.2	8.5	0.8
1	01:00	3	7.1	2.2	0.2
2	01:05	1	3.4	7.2	0.7
2	01:05	2	5.2	8.5	0.9
2	01:05	3	7.1	2.2	1.0

sensor_values

Figure 2.2. Database table containing the sensor values. The position of the sensor s_j is denoted as (x_j, y_j). Since the sensors are assumed to be stationary, the position can also be stored using a foreign-key relationship between s_j and (x_j, y_j). But, for simplicity, we assume that the `sensor_values` table is in a denormalized form.

In this chapter, we assume a scenario where the sensors are used for environmental monitoring. We assume that all the sensors are monitoring/sensing only one environmental attribute, such as, ambient temperature[2]. As discussed in Section 1, we assume that the environmental attribute we monitor is sufficiently smooth and continuous. If necessary for rendering the discussion complete and convenient, we will introduce other attributes being monitored by the sensors. But, in most cases, we restrict ourselves to using only ambient temperature. Figure 2.2 shows a conceptual representation of the sensor values in a form of a database table, denoted as `sensor_values`.

2.2 The Sensor Data Acquisition Query

Sensor data acquisition can be defined as the processes of creating and continuously maintaining the `sensor_values` table. In existing literature, naturally, many techniques have been proposed for creating and maintaining the `sensor_values` table. We shall discuss these techniques briefly, describing their important characteristics and differences with other techniques. We use the sensor data acquisition query shown in Query 2.1 for discussing how different sensor data acquisition approaches process such a query. Query 2.1 is a query that triggers the acquisition of ten sensor values v_{ij} from the sensors s_j at a sampling interval of one second. Moreover, Query 2.1, is the typical sensor data acquisition query that is used by many methods for collecting sensor data.

[2]We use *ambient temperature* and *temperature* interchangeably.

```
SELECT s_j, v_ij FROM sensor_values SAMPLE INTERVAL 1s FOR 10s
```

Query 2.1: Sensor data acquisition query.

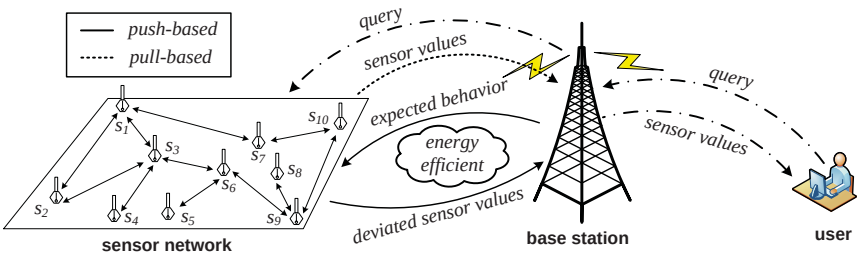


Figure 2.3. Push- and pull-based methods for sensor data acquisition.

2.3 Pull-Based Data Acquisition

Broadly, there are two major approaches for data acquisition: pull-based and push-based (refer Figure 2.3). In the pull-based sensor data acquisition approach, the user defines the interval and frequency of data acquisition. Pull-based systems only follow the user's requirements, and pull sensor values as defined by the queries. For example, using the `SAMPLE INTERVAL` clause of Query 2.1, users can specify the number of samples and the frequency at which the samples should be acquired.

2.3.1 In-Network Data Acquisition. This approach of sensor data acquisition is proposed by TinyDB [45, 44, 43], Cougar [69] and TiNA [58]. These approaches tightly link query processing and sensor data acquisition. Due to the lack of space, we shall only discuss TinyDB in this subsection.

TinyDB refers to its in-network query processing paradigm as *Acquisitional Query Processing* (ACQP). Let us start by discussing how ACQP processes Query 2.1. The result of Query 2.1 is similar to the table shown in Figure 2.2. The only difference, as compared to Figure 2.2, is that the result of Query 2.1 contains $10 \times m$ rows. The naïve method of executing Query 2.1 is to simultaneously poll each sensor for its value at the sampling interval and for the duration specified by the query. This method may not work due to limited range of radio communication between individual sensors and the base station.

Data Acquisition using Semantic Overlays: TinyDB proposes a tree-based overlay that is constructed using the sensors **S**. This tree-based overlay is used for aggregating the query results from the leaf nodes to the root node. The overlay network is especially built for efficient data acquisition and query processing. TinyDB refers to its

tree-based overlay network as *Semantic Routing Trees* (SRTs). A SRT is constructed by flooding the sensor network with the *SRT build request*. This request includes the attribute (ambient temperature), over which the SRT should be constructed. Each sensor s_j, which receives the build request, has several choices for choosing its parent: (a) if s_j has no children, which is equivalent to saying that no other sensor has chosen s_j as its parent, then s_j chooses another sensor as its parent and sends its current value v_{ij} to the chosen parent in a *parent selection message*, or (b) if s_j has children, it sends a parent selection message to its parent indicating the range of ambient temperature values that its children are covering. In addition, it locally stores the ambient temperature values from its children along with their sensor identifiers.

Next, when Query 2.1 is presented to the root node of the SRT, it forwards the query to its children and prepares for receiving the results. At the same time, the root node also starts processing the query locally (refer Figure 2.4). The same procedure is followed by all the intermediate sensors in the SRT. A sensor that does not have any children, processes the query and forwards the value of v_{ij} to its parent. All the collected sensor values v_{ij} are finally forwarded to the root node, and then to the user, as a result of the query. This completes the processing of the sensor data acquisition query (Query 2.1). The SRT, moreover, can also be used for optimally processing aggregation, threshold, and event based queries. We shall return to this point later in Section 4.1.

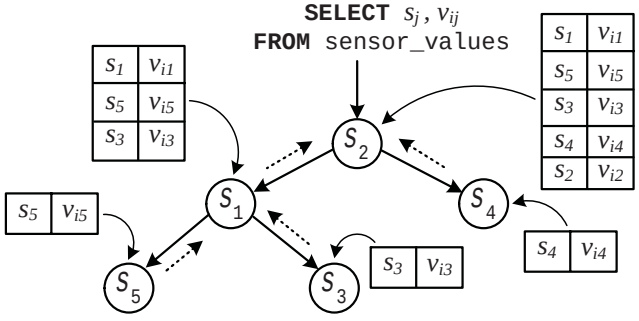


Figure 2.4. Toy example of a Semantic Routing Tree (SRT) and Acquisitional Query Processing (ACQP) over a sensor network with five sensors. Dotted arrows indicate the direction of query response. A given sensor appends its identifier s_j and value v_{ij} to the partial result, which is available from its sub-tree.

2.3.2 Multi-Dimensional Gaussian Distributions. The Barbie-Q (BBQ) system [17, 16], on the other hand, employs multivariate Gaussian distributions for sensor data acquisition. BBQ maintains a multi-dimensional Gaussian probability distribution over all the

sensors in $\mathbf{S}$. Data is acquired only as much as it is required to maintain such a distribution. Sensor data acquisition queries specify certain confidence that they require in the acquired data. If the confidence requirement cannot be satisfied, then more data is acquired from the sensors, and the Gaussian distribution is updated to satisfy the confidence requirements. The BBQ system models the sensor values using a multi-variate Gaussian probability density function (pdf) denoted as $p(V_{i1}, V_{i2}, \ldots, V_{im})$, where $V_{i1}, V_{i2}, \ldots, V_{im}$ are the random variables associated with the sensor values $v_{i1}, v_{i2}, \ldots, v_{im}$ respectively. This pdf assigns a probability for each possible assignment of the sensor values v_{ij}. Now, let us discuss how the BBQ system processes Query 2.1.

In BBQ, the inferred sensor value of sensor s_j, at each time t_i, is defined as the mean value of V_{ij}, and is denoted as $\bar{v}_{ij}$. For example, at time t_1, the inferred sensor values of the ambient temperature are $\bar{v}_{11}, \bar{v}_{12}, \ldots, \bar{v}_{1m}$. The BBQ system assumes that queries, like Query 2.1, provide two additional constraints: (i) error bound ϵ, for the values $\bar{v}_{ij}$, and (ii) the confidence $1 - \delta$ with which the error bound should be satisfied. Admittedly, these additional constraints are for controlling the quality of the query response.

Suppose, we already have a pdf before the first time instance t_1, then the confidence of the sensor value v_{1j} is defined as the probability of the random variable V_{1j} lying in between $\bar{v}_{1j} - \epsilon$ and $\bar{v}_{1j} + \epsilon$, and is denoted as $P(V_{1j} \in [\bar{v}_{1j} - \epsilon, \bar{v}_{1j} + \epsilon])$. If the confidence is greater than $1 - \delta$, then we can provide a probably approximately correct value for the temperature, without spending energy in obtaining a sample from sensor s_j. On the other hand, if a sensor's confidence is less than $1 - \delta$, then we should obtain one or more samples from the sensor (or other correlated sensors), such that the confidence bound is satisfied. In fact, it is clear that there could be potentially many sensors for which the confidence bound may not hold.

As a solution to this problem, the BBQ system proposes a procedure to chose the sensors for obtaining sensor values, such that the confidence bound specified by the query is satisfied. First, the BBQ system samples from all the sensors $\mathbf{S}$ at time t_1, then it computes the confidence $\mathcal{B}_j(\mathbf{S})$ that it has in a sensor s_j as follows:

$$\mathcal{B}_j(\mathbf{S}) = P(V_{1j} \in [\bar{v}_{1j} - \epsilon, \bar{v}_{1j} + \epsilon] | v_1), \tag{2.1}$$

where $v_1 = (v_{11}, v_{12}, \ldots, v_{1m})$ is the row vector of all the sensor values at time t_1. Second, for choosing sensors to sample, the BBQ system poses an optimization problem of the following form:

$$\min_{\mathbf{S}_o \subseteq \mathbf{S} \text{ and } \mathcal{B}(\mathbf{S}_o) \geq 1-\delta.} \mathcal{C}(\mathbf{S}_o), \tag{2.2}$$

where $\mathbf{S}_o$ is the subset of sensors that will be chosen for sampling, $\mathcal{C}(\mathbf{S}_o)$ and $\mathcal{B}(\mathbf{S}_o) = \frac{1}{|\mathbf{S}_o|} \sum_{j:s_j \in \mathbf{S}_o} \mathcal{B}_j(\mathbf{S})$ are respectively the total cost (or energy required) and average confidence for sampling sensors $\mathbf{S}_o$. Since the problem in Eq. (2.2) is NP-hard, BBQ proposes a greedy solution to solve this problem. Details of this greedy algorithm can be found in [17]. By executing the proposed greedy algorithm, BBQ selects the sensors for sampling, then it updates the Gaussian distribution, and returns the mean values $\bar{v}_{11}, \bar{v}_{12}, \ldots, \bar{v}_{1m}$. These mean values represent the inferred values of the sensors at time t_1. This operation when performed ten times at an interval of one second generates the result of the sensor data acquisition query (Query 2.1).

2.4 Push-Based Data Acquisition

Both, TinyDB and BBQ, are pull-based in nature: in these systems the central server/base station decides when to acquire sensor values from the sensors. On the other hand, in push-based approaches, the sensors autonomously decide when to communicate sensor values to the base station (refer Figure 2.3). Here, the base station and the sensors agree on an expected behavior of the sensor values, which is expressed as a model. If the sensor values deviate from their expected behavior, then the sensors communicate only the deviated values to the base station.

2.4.1 PRESTO. The Predictive Storage (PRESTO) [41] system is an example of the push-based data acquisition approach. One of the main arguments that PRESTO makes against pull-based approaches is that due to the pull strategy, such approaches will be unable to observe any unusual or interesting patterns between any two pull requests. Moreover, increasing the pull frequency for better detection of such patterns, increases the overall energy consumption of the system.

The PRESTO system contains two main components: PRESTO proxies and PRESTO sensors. As compared to the PRESTO sensors, the PRESTO proxies have higher computational capability and storage resources. The task of the proxies is to gather data from the PRESTO sensors and to answer queries posed by the user. The PRESTO sensors are assumed to be battery-powered and remotely located. Their task is to sense the data and transmit it to PRESTO proxies, while archiving some of it locally on flash memory.

Now, let us discuss how PRESTO processes the sensor data acquisition query (Query 2.1). For answering such a query, the PRESTO proxies always maintain a time-series prediction model. Specifically, PRESTO maintains a seasonal ARIMA (SARIMA) model [60] of the

following form for each sensor:

$$v_{ij} = v_{(i-1)j} + v_{(i-L)j} - v_{(i-L-1)j} + \theta e_{i-1} - \Theta e_{i-L} + \theta\Theta e_{i-L-1}, \quad (2.3)$$

where θ and Θ are parameters of the SARIMA model, e_i are the prediction errors and L is known as the seasonal period. For example, while monitoring temperature, L could be set to one day, indicating that the current temperature (v_{ij}) is related to the temperature yesterday at the same time ($v_{(i-L)j}$) and a previous time instant ($v_{(i-L-1)j}$). In short, the seasonal period L allows us to model the periodicity that is inherent in certain types of data.

In the PRESTO system the proxies estimate the parameters of the model given in Eq. (2.3), and then transmit these parameters to individual PRESTO sensors. The PRESTO sensors use these models to predict the sensor value $\hat{v}_{ij}$, and only transmit the raw sensor value v_{ij} to the proxies when the absolute difference between the predicted sensor value and the raw sensor value is greater than a user-defined threshold δ. This task can be summarized as follows:

$$|v_{ij} - \hat{v}_{ij}| > \delta, \text{ transmit } v_{ij} \text{ to proxy.} \quad (2.4)$$

The PRESTO proxy also provides a confidence interval for each predicted value it computes using the SARIMA model. Like BBQ (refer Section 2.3.2), this confidence interval can also be used for query processing, since it represents an error bound on the predicted sensor value. Similar to BBQ, the PRESTO proxy queries the PRESTO sensors only when the desired confidence interval, specified by the query, could not be satisfied with the values stored at the PRESTO proxy. In most cases, the values stored at the proxy can be used for query processing, without acquiring any further values from the PRESTO sensors. The only difference between PRESTO and BBQ is that, PRESTO uses a different measure of confidence as compared to BBQ. Further details of this confidence interval can be found in [41].

2.4.2 Ken. For reducing the communication cost, the Ken [12] framework employs a similar strategy as PRESTO. Although there is a key difference between Ken and PRESTO. PRESTO uses a SARIMA model; this model only takes into account temporal correlations. On the other hand, Ken uses a dynamic probabilistic model that takes into account spatial and temporal correlations in the data. Since a large quantity of sensor data is correlated spatially, and not only temporally, Ken derives advantage from such spatio-temporal correlation.

The Ken framework has two types of entities, *sink* and *source*. Their functionalities and capabilities are similar to the PRESTO proxy and the

PRESTO sensor respectively. The only difference is that the PRESTO sensor only represents a single sensor, but a source could include more than one sensor or a sensor network. The sink is the base station to which the sensor values v_{ij} are communicated by the source (refer Figure 2.3).

The fundamental idea behind Ken is that both, source and sink, maintain the same dynamic probabilistic model of data evolution. The source only communicates with the sink when the raw sensor values deviate beyond a certain bound, as compared to the predictions from the dynamic probabilistic model. In the meantime, the sink uses the sensor values predicted by the model.

As discussed before, Ken uses a dynamic probabilistic model that considers spatio-temporal correlations. Particularly, its dynamic probabilistic model computes the following pdf at the source:

$$p(V_{(i+1)1}, \ldots, V_{(i+1)m} | v_1, \ldots, v_i) = \int p(V_{(i+1)1}, \ldots, V_{(i+1)m} | V_{i1}, \ldots, V_{im}) \\ p(V_{i1}, \ldots, V_{im} | v_1, \ldots, v_i) dV_{i1} \ldots dV_{im}. \tag{2.5}$$

This pdf is computed using the observations that have been communicated to the sink; the values that are not communicated to the sink are ignored by the source, since they do not affect the model at the sink. Next, each sensor contained in the source computes the expected sensor value using Eq. (2.5) as follows:

$$\bar{v}_{(i+1)j} = \int V_{(i+1)j} p(V_{(i+1)1}, \ldots, V_{(i+1)m}) dV_{(i+1)1} \ldots dV_{(i+1)m}. \tag{2.6}$$

The source does not communicate with the sink if $|\bar{v}_{(i+1)j} - v_{(i+1)j}| < \delta$, where δ is a user-defined threshold. If this condition is not satisfied, the source communicates to the sink the smallest number of sensor values, such that the δ threshold would be satisfied. Similarly, if the sink does not receive any sensor values from the source, it computes the expected sensor values $\bar{v}_{(i+1)j}$ and uses them as an approximation to the raw sensor values. If the sink receives a few sensor values form the source, then, before computing the expected values, the sink updates its dynamic probabilistic model.

2.4.3 A Generic Push-Based Approach. The last push-based approach that we will survey is a generalized version of other push-based approaches [38]. This approach is proposed by Silberstein *et al.* [61]. Like other push-based approaches, the base station and the sensor network agree on an expected behavior, and, as usual, the sensor network reports values only when there is a substantial deviation from

the agreed behavior. But, unlike other approaches, the definition of expected behavior proposed in [61] is more generic, and is not limited to a threshold δ.

In this approach a sensor can either be an updater (one who acquires or forwards sensor values) or an observer (one who receives sensor values). A sensor node can be both, updater and observer, depending on whether it is on the boundary of the sensor network or an intermediate node. The updaters and the observers maintain a model encoding function f_{enc} and a decoding function f_{dec}. These model encoding/decoding functions define the agreed behavior of the sensor values. The updater uses the encoding function to encode the sensor value v_{ij} into a transmission message g_{ij}, and transmits it to the observer.

The observer, then, uses the decoding function f_{dec} to decode the message g_{ij} and construct $\hat{v}_{ij}$. If the observer finds that v_{ij} has not changed significantly, as defined by the encoding function, then the observer transmits a `null` symbol. A `null` symbol indicates that the sensor value is *suppressed* by the observer. Following is an example of the encoding and decoding functions [61]:

$$f_{enc}(v_{ij}, v_{i'j}) = \begin{cases} g_{ij} = v_{ij} - v_{i'j}, & \text{if } |v_{ij} - v_{i'j}| > \delta; \\ g_{ij} = \texttt{null}, & \text{otherwise.} \end{cases} \tag{2.7}$$

$$f_{dec}(g_{ij}, \hat{v}_{(i-1)j}) = \begin{cases} \hat{v}_{(i-1)j} + g_{ij}, & \text{if } g_{ij} \neq \texttt{null}; \\ \hat{v}_{(i-1)j}, & \text{if } g_{ij} = \texttt{null}. \end{cases} \tag{2.8}$$

In the above example, the encoding function f_{enc} computes the difference between the model predicted sensor value $v_{i'j}$ and the raw sensor value v_{ij}. Then, this difference is transmitted to the observer only if it is greater than δ, otherwise the `null` symbol is transmitted. The decoding function f_{dec} decodes the sensor value $\hat{v}_{(i-1)j}$ using the message g_{ij}.

The encoding and decoding functions in the above example are purposefully chosen to demonstrate how the δ threshold approach can be replicated by these functions. More elaborate definitions of these functions, which are used for encoding complicated behavior, can be found in [61].

3. Model-Based Sensor Data Cleaning

A well-known characteristic of sensor data is that it is uncertain and erroneous. This is due to the fact that sensors often operate with discharged batteries, network failures, and imprecision. Other factors, such as low-cost sensors, freezing or heating of the casing or measurement device, accumulation of dirt, mechanical failure or vandalism (from hu-

mans or animals) heavily affect the quality of the sensor data [31, 73, 23]. This may cause a significant problem with respect to data utilization, since applications using erroneous data may yield unsound results. For example, scientific applications that perform prediction tasks using observation data obtained from cheap and less-reliable sensors may produce inaccurate prediction results.

To address this problem, it is essential to detect and correct erroneous values in sensor data by employing *data cleaning*. The data cleaning task typically involves complex processing of data [71, 30]. In particular, it becomes more difficult for sensor data, since true sensor values corresponding to erroneous data values are generally unobservable. This has led to a new approach – *model-based data cleaning*. In this approach, the most probable sensor values are inferred using well-established models, and then anomalies are detected by comparing raw sensor values with the corresponding inferred sensor values. In the literature there are a variety of suggestions for model-based approaches for sensor data cleaning. This section describes the key mechanisms proposed by these approaches, particularly focusing on the models used in the data cleaning process.

3.1 Overview of Sensor Data Cleaning System

A system for cleaning sensor data generally consists of four major components: *user interface, stream processing engine, anomaly detector*, and *data storage* (refer Figure 2.5). In the following, we describe each component.

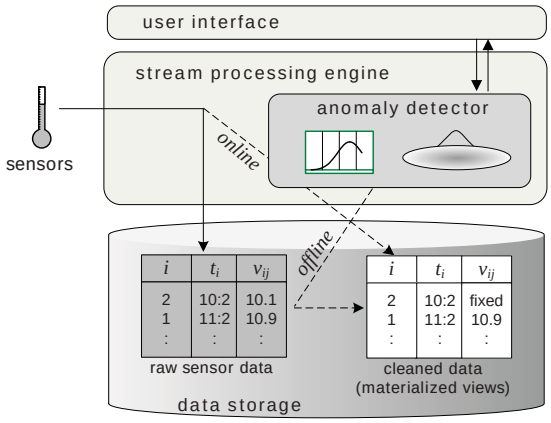


Figure 2.5. Architecture of sensor data cleaning system.

User Interface: The user interface plays two roles in the data cleaning process. First, it takes all necessary inputs from users to perform data cleaning, e.g., name of sensor data and parameter settings for models. Second, the results of data cleaning, such as 'dirty' sensor values captured by the anomaly detector, are presented using graphs and tables, so that users can confirm whether each candidate of such dirty values is an actual error. The confirmed results are then stored to (or removed from) the underlying data storage or materialized views.

Anomaly Detector: The anomaly detector is a core component in sensor data cleaning. It uses models for detecting abnormal data values. The anomaly detector works in online as well as offline mode. In the online mode, whenever a new sensor value is delivered to the stream processing engine, the dirtiness of this value is investigated and the errors are filtered out instantly. In the offline mode, the data is cleaned periodically, for instance, once per day. In the following subsections, we will review popular models used for online anomaly detection.

Stream Processing Engine: The stream processing engine maintains streaming sensor data, while serving as a main platform where the other system components can cooperatively perform data cleaning. The anomaly detector is typically embedded into the stream processing engine, it may also be implemented as a built-in function on database systems.

Data Storage: The data storage maintains not only sensor values, but also the corresponding cleaned data, typically in materialized views. This is because applications on sensor networks often need to repeatedly perform data cleaning over the same data using different parameter settings for the models, especially when the previous parameter settings turn out to be inappropriate later. Therefore, it is important for the system to store cleaned data in database views without changing the original data, so that data cleaning can be performed again at any point of time (or time interval) as necessary.

3.2 Models for Sensor Data Cleaning

This subsection reviews popular models that are widely used in the sensor data cleaning process.

3.2.1 Regression Models. As sensor values are a representation of physical processes, it is naturally possible to uncover the following properties: continuity of the sampling processes and correlations between different sampling processes. In principle, regression-based models

exploit either or both of these properties. Specifically, they first compute the dependency from one variable (e.g., time) to another (e.g., sensor value), and then consider the regression curves as standards over which the inferred sensor values reside. The two most popular regression-based approaches use polynomial and Chebyshev regression for cleaning sensor values.

Polynomial Regression: Polynomial regression finds the best-fitting curve that minimizes the total difference between the curve and each raw sensor value v_{ij} at time t_i. Given a degree d, polynomial regression is formally defined as:

$$v_{ij} = c + \alpha_1 \cdot t_i + \cdots + \alpha_d \cdot t_i^d, \tag{2.9}$$

where c is a constant and $\alpha_1, \ldots, \alpha_d$ are regression coefficients.

Polynomial regression with high degrees approximate given time series with more sophisticated curves, resulting in theoretically more accurate description of the raw sensor values. Practically, however, low-degree polynomials, such as constant ($d = 0$) and linear ($d = 1$), also perform satisfactorily. In addition, low-degree polynomials can be more efficiently constructed as compared to high-degree polynomials. A (weighted) moving average model [73] is also regarded as a polynomial regression.

Chebyshev Regression: Chebyshev regression is a popular model class for fitting sensor values, since they can quickly compute near-optimal approximations for given time series. Suppose that time values t_i vary within a range $[\min(t_i), \max(t_i)]$. We, then, obtain normalized time values t_i' within a range $[-1, 1]$, by using the following transformation function $f(t_i)$ and its inverse transformation function $f^{-1}(t_i')$ as follows:

$$f(t_i) = \left(t_i - \frac{\max(t_i) + \min(t_i)}{2}\right) \cdot \frac{2}{\max(t_i) - \min(t_i)}, \tag{2.10}$$

$$f^{-1}(t_i') = \left(t_i' \cdot \frac{\max(t_i) - \min(t_i)}{2}\right) + \frac{\max(t_i) + \min(t_i)}{2}. \tag{2.11}$$

Next, given a degree d, Chebyshev polynomial is defined as:

$$v_{ij} = f^{-1}(\cos(d \cdot \cos^{-1}(f(t_i)))).$$

Figure 2.6 illustrates a data cleaning process using degree-2 Chebyshev polynomials. Here, the raw sensor values are plotted as green curves, while the inferred values, obtained by fitting a Chebyshev polynomials, are overlaid by black curves. The anomaly points are then indicated by the underlying red histograms as well as red circles.

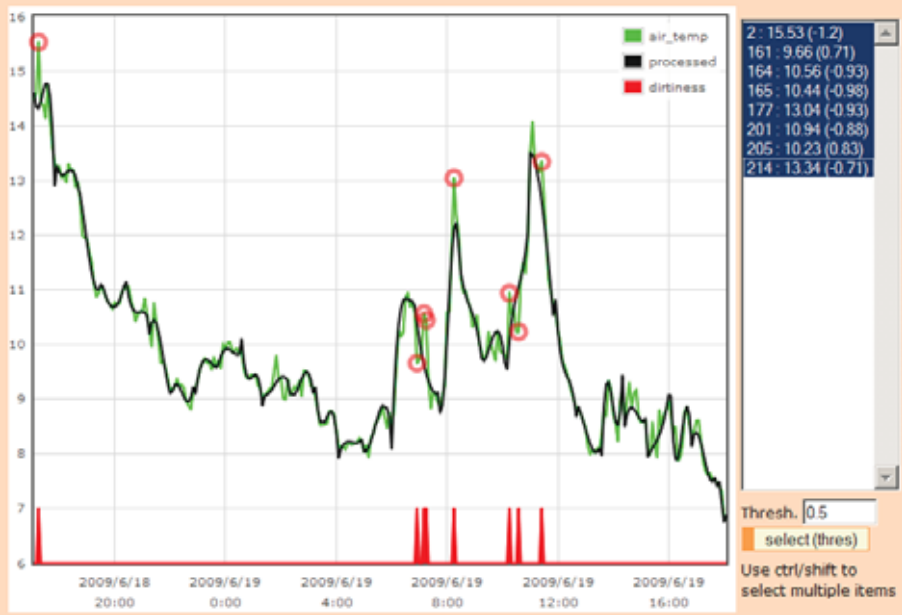


Figure 2.6. Detected anomalies based on 2-degree Chebyshev regression.

3.2.2 Probabilistic Models. In sensor data cleaning, inferring sensor values is perhaps the most important task, since systems can then detect and clean dirty sensor values by comparing raw sensor values with the corresponding inferred sensor values. Figure 2.7 shows an example of the data cleaning process using probabilistic models. At time $t_i = 6$, the probabilistic model infers a probability distribution using the previous values $v_{2j}, \ldots, v_{5j}$ in the sliding window. The expected value $\bar{v}_{6j}$ (e.g., the mean of the Gaussian distribution in the future) is then considered as the inferred sensor value for sensor s_j.

Next, the anomaly detector checks whether the raw sensor value v_{6j} resides within a reasonably accurate area. This is done in order to check whether the value is *normal*. For instance, the 3σ range can cover 99.7 % of the density in the figure, where v_{6j} is supposed to appear. Thus, the data cleaning process can consider that v_{6j} is not an error. At $t_i = 7$, the window slides and now contains raw sensor values $v_{3j}, \ldots, v_{6j}$. By repeating the same process, the anomaly detector finds v_{7j} resides out of the error bound (3σ range) in the inferred probability distribution, and is identified as an anomaly [57].

A vast body of research work has utilized probabilistic models for computing inferred values. The *Kalman filter* is perhaps one of the most

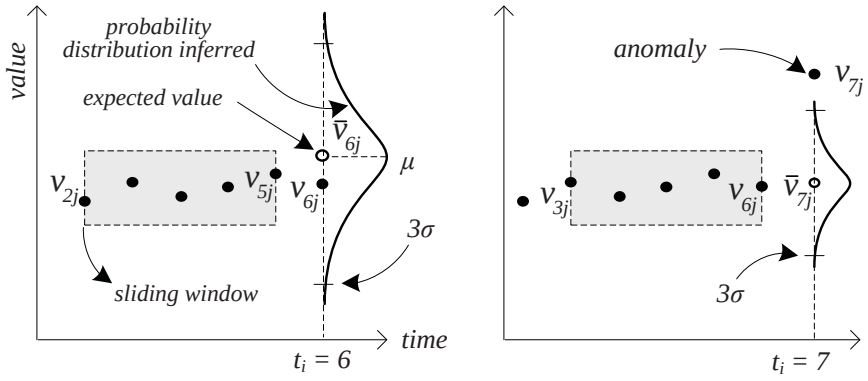


Figure 2.7. An example of data cleaning based on a probabilistic model.

common probabilistic models to compute inferred values corresponding to raw sensor values. The Kalman filter is a stochastic and recursive data filtering algorithm that models the raw sensor value v_{ij} as a function of its previous value (or state) $v_{(i-1)j}$ as follows:

$$v_{ij} = Av_{(i-1)j} + Bu_i + w_i,$$

where A and B are matrices defining the state transition from time t_{i-1} to time t_i, u_i is the time-varying input at time t_i, and w_i is the process noise drawn from a zero mean multi-variate Gaussian distribution. In [63], the Kalman filter is used for detecting erroneous values, as well as inter/extrapolating missing sensor values. Jain *et al.* [29] also use the Kalman filter for filtering possible dirty values.

Similarly, Elnahrawy and Nath [21] proposed to use Bayes' theorem to estimate a probability distribution P_{ij} at time t_i from raw sensor values v_{ij}, and associate them with an error model, typically a normal distribution. Built on the same principle, a neuro-fuzzy regression model [52] and a belief propagation model based on Markov chains [13] were used to identify anomalies. Tran *et al.* [65] propose a method to infer missing or erroneous values in RFID data. All the techniques for inferring sensor values also enable quality-aware processing of sensor data streams [36, 37], since inferred sensor values can serve as the bases for indicating the quality or precision of the raw sensor values.

3.2.3 Outlier Detection Models. An *outlier* is a sensor value that largely deviates from the other sensor values. Obviously, outlier detection is closely related to the process of sensor data cleaning. The outlier-detection techniques are well-categorized in the survey studies of [51, 8].

In particular, some of the outlier detection methods focus on sensor data [59, 71, 15]. Zhang *et al.* [71] offer an overview of such outlier detection techniques for sensor network applications. Deligiannakis *et al.* [15] consider correlation, extended Jaccard coefficients, and regression-based approximation for model-based data cleaning. Shen *et al.* [59] propose to use a histogram-based method to capture outliers. Subramaniam *et al.* [62] introduce distance- and density-based metrics that can identify outliers. In addition, the ORDEN system [23] detects polygonal outliers using the triangulated wireframe surface model.

3.3 Declarative Data Cleaning Approaches

From the perspective of using a data cleaning system, supporting a declarative interface is important since it allows users to easily control the system. This idea is reflected in a wide range of prior work that proposes SQL-like interfaces for data cleaning [32, 46, 54]. These proposals hide complicated mechanisms of data processing or model utilization from the users, and facilitate data cleaning in sensor network applications.

More specifically, Jeffery *et al.* [31, 32] divide the data cleaning process into five tasks: *Point, Smooth, Merge, Arbitrate*, and *Virtualize*. These tasks are then supported within a database system. For example, the SQL statement in Query 2.2 performs anomaly detection within a spatial granule by determining the average of the sensor values from different sensors in the same proximity group. Then, individual sensor values are rejected if they are outside of one standard deviation from the mean.

As another approach, Rao *et al.* [54] focus on a systemic solution, based on rewriting queries using a set of cleansing rules. Specifically, the system offers the rule grammar shown in Figure 2.8 to define and execute various data cleaning tasks. Unlike the prior relational database approaches, Mayfield *et al.* [46] model data as a graph consisting of nodes and links. They, then, provide an SQL-based, declarative frame-

```
DEFINE        [rule name]
ON            [table name]
FROM          [table name]
CLUSTER BY    [cluster key]
SEQUENCE BY   [sequence key]
AS            [pattern]
WHERE         [condition]
ACTION        [DELETE | MODIFY | KEEP]
```

Figure 2.8. An example of anomaly detection using a SQL statement.

```
SELECT spatial_granule, AVG(temp)
FROM data s [Range By 5 min]
       (SELECT spatial_granule, avg(temp) as avg,
       stdev(temp) as stdev
       FROM data [Range By 5 min]) as a
WHERE a.spatial_granule = s.spatial_granule
       AND a.avg + (2*a.stdev) < s.temp
       AND a.avg - (2*a.stdev) > s.temp
```

Query 2.2: An example of anomaly detection using a SQL statement.

work that enables data owners to specify or discover groups of attributes that are correlated, and apply statistical methods that validate and clean the sensor values using such dependencies.

4. Model-Based Query Processing

In this section we elaborate another important task in sensor data management – query processing. We primarily focus on in-network and centralized query processing approaches. We consider different queries assuming the sensor network described in Section 2.1, and then discuss how each approach processes these queries. In Section 2, however, we followed an approach where we chose a singe query (i.e., Query 2.1) and demonstrated how different techniques processed this query. On the contrary, in this section, we chose different queries for all the approaches, and then discuss these approaches along with the queries. We follow this procedure since, unlike Section 2, the assumptions made by each query processing technique are different. Thus, for highlighting the impact of these assumptions and simplifying the discussion, we select different queries for each approach.

4.1 In-Network Query Processing

In-network query processing first builds an overlay network (like, the SRT discussed in Section 2.3.1). Then, the overlay network is used for increasing the efficiency of aggregating sensor values and processing queries. For instance, while processing a threshold query, parent nodes send the query to the child nodes only when the query threshold condition overlaps with the range of sensor values contained in the child nodes, which is stored in the parent node's local memory.

Consider the threshold query given in Query 2.3. Query 2.3 requests the sensor identifiers of all the sensors that have sensed a temperature greater than 10°C at the current time instance. Before answering this query, we assume that we have already constructed a SRT as described

in Section 2.2 (refer Figure 2.4). Query 2.3 is sent by the root node of the SRT to its children that are a part of the query response. The child nodes check whether the sensor value they have sensed is greater than 10°C. If the sensor value is greater than 10°C at a child node, then that child node appends its sensor identifier to the query response. The child node, then, forwards the query to its children and waits for their response. Once all the children of a particular node have responded, then that node forwards the response of its entire sub-tree to its parent. In the end, the root node receives all the sensor identifiers s_j that have recorded temperature greater than 10°C.

```
SELECT s_j FROM sensor_values WHERE v_ij > 10°C AND t_i == NOW()
```

Query 2.3: Return the sensor identifiers s_j where $v_{ij} > 10°C$.

4.2 Model-Based Views

The MauveDB [18] approach proposes standard database views [19] as an abstraction layer for processing queries. These views are maintained in a form of a regression model; thus they are called *model-based* views. The main advantage of this approach is that the model-based view can be incrementally updated as fresh sensor values are obtained from the sensors. Furthermore, incremental updates is an attractive feature, since such updates are computationally efficient.

Before processing any queries in MauveDB, we have to first create a model-based view. The query for creating a model-based view is shown in Query 2.4. The model-based view created by this query is called `RegModel`. `RegModel` is a regression model in which the temperature is the dependent variable and the sensor position (x_j, y_j) is an independent variable (refer Figure 2.9). Note that `RegModel` is incrementally updated by MauveDB. At time t_1 values from sensors s_1, s_3 and at time t_2 the value from sensor s_2 are respectively used to update the view. The view update mechanism exploits the fact that regression functions can be updated. Further details regarding the update mechanism can be found in [18].

```
CREATE VIEW RegModel AS FIT v OVER x^2, xy, y^2, x, y
TRAINING_DATA SELECT x_j, y_j, v_ij FROM sensor_values
WHERE t_i > t_start AND t_i < t_end
```

Query 2.4: Model-based view creation query.

Once this step is performed many types of queries can be evaluated using the `RegModel` view. For instance, consider Query 2.5. MauveDB

evaluates this query by interpolating the value of temperature at fixed intervals on the x- and y-axis; this is similar to database view materialization [19]. Then the positions (x, y) where the interpolated temperature value is greater than 10°C are returned.

Admittedly, although updating the model-based view is efficient, but for processing queries the model-based view should be materialized at a certain fixed set of points. This procedure produces a large amount of overhead when the number of independent variables is large, since it dramatically increases the number of points where the view should be materialized.

```
SELECT x, y FROM RegModel WHERE v > 10°C
```

Query 2.5: Querying model-based views.

4.3 Symbolic Query Evaluation

This approach is proposed by the FunctionDB [64] system. FunctionDB, like MauveDB, also interpolates the values of the dependent variable, and then uses the interpolated values for query processing.

As discussed before, the main problem with value interpolation is that the number of points, where the sensor values should be interpolated, increase dramatically as a function of the number of independent variables. As a solution to this problem, FunctionDB symbolically executes the filter (for example, the `WHERE` clause in Query 2.5) and obtains feasible regions of the independent variables. These feasible regions are the regions that include the exact response to the query, at the same time contain a significantly low number of values to interpolate. FunctionDB

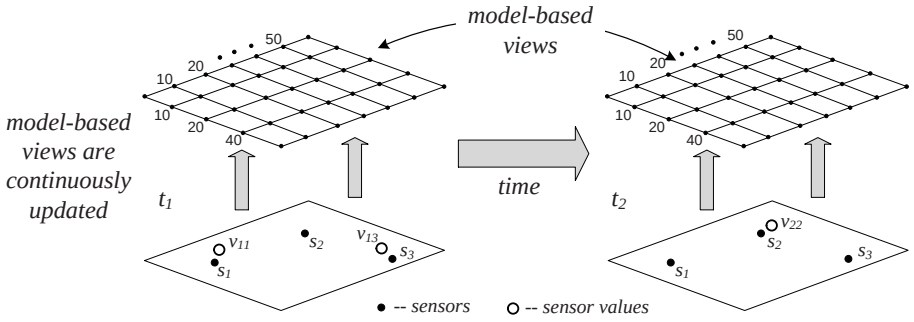


Figure 2.9. Example of the `RegModel` view with three sensors. `RegModel` is incrementally updated as new sensor values are acquired.

evaluates the query by interpolating values only in the feasible regions, followed by a straightforward evaluation of the query.

Moreover, FunctionDB treats the temperature of the sensor s_j as a continuous function of time $f_j(t)$, instead of treating it as discrete values sampled at time stamps t_i. An example of a query in the FunctionDB framework is given in Query 2.6. This query returns the time values t between t_{start} and t_{end} where the temperature of the sensor s_1 is greater than 10°C. Note that the time values t are not necessarily the time stamps t_i where a particular sensor value was recorded.

`SELECT` t `WHERE` $f_1(t) > 10^{\circ}\text{C}$ `AND` $t > t_{start}$ `AND` $t < t_{end}$ `GRID t 1s`

Query 2.6: Continuous threshold query.

For defining the values of the time axis t (or any continuous variable), FunctionDB proposes the `GRID` operator. The `GRID` operator specifies the interval at which the function $f_1(t)$ should be interpolated between time t_{start} and t_{end}. For instance, `GRID t 1s` indicates that the time axis should be interpolated at one second intervals between time t_{start} and t_{end}. To process Query 2.6, FunctionDB first symbolically executes the `WHERE` clause and obtains the feasible regions of the time axis (independent variable). Then, using the `GRID` operator, it generates time stamps T_I in the feasible regions. The sensor value is interpolated at the time stamps T_I using regression functions. Lastly, the query is processed on these interpolated values, and time stamps $T'_I \subseteq T_I$ where the temperature is greater than 10°C are returned.

4.4 Processing Queries over Uncertain Data

In this form of query processing the assumption is that sensor data is inherently uncertain. This uncertainty can arise due to various factors: loss of calibration over time, faulty sensors, unsuitable environmental conditions, low sensor accuracy, etc. Thus, the approaches that treat sensor data as uncertain, assume that each sensor value is associated with a random variable, and is drawn from a distribution. In this subsection, we discuss two such methods that model uncertain data by either a dynamic probabilistic model or a static probability distribution.

4.4.1 Dynamic Probabilistic Models. Dynamic probabilistic models (DPMs) are proposed for query processing in [33, 29]. These models continuously estimate a probability distribution. The estimated probability distribution is used for query processing. Mainly, there are two types of models that are frequently used for estimating dynamic probability distributions: particle filters and Kalman filters. Particle

filters are generalized form of Kalman filters. Since we have already discussed Kalman filters in Section 3.2, here we will focus on particle filtering.

Consider a single sensor, say s_1, the particle filtering approach [4], at each time instant t_i, estimates and stores p weighted tuples $\{(w_{i1}^1, v_{i1}^1), \ldots, (w_{i1}^p, v_{i1}^p)\}$, where the weight w_{i1}^1 denotes the probability of v_{i1}^1 being the sensor value of the sensor s_1 at time t_i, and so on. An example of particle filtering is shown in the `pf_sensor_values` table in Figure 2.10.

Now, consider the Query 2.7 that requests the average temperature $\texttt{AVG}(v_{ij})$ between time t_{start} and t_{end}. To evaluate this query, we assume that we already have executed the particle filtering algorithm at each time instance t_i and have created the `pf_sensor_values` table. We, then, perform the following two operations:

1. For each time t_i between t_{start} and t_{end}, we compute the expected temperature $\bar{v}_{i1} = \sum_{l=1}^{p} w_{i1}^l \cdot v_{i1}^l$. The formal SQL syntax for computing the expected values using the `pf_sensor_values` table is as follows:

   `SELECT` $t_i, \sum_{l=1}^{p} w_{i1}^l \cdot v_{i1}^l$ `FROM pf_sensor_values WHERE` $t_i > t_{start}$ `AND` $t_i < t_{end}$ `GROUP BY` t_i

2. The final result is the average of all the $\bar{v}_{i1}$ that we computed in Step 1.

Essentially, the tuples $\{(w_{i1}^1, v_{i1}^1), \ldots, (w_{i1}^p, v_{i1}^p)\}$ represent a discretized pdf for the random variable V_{i1}. Moreover, the most challenging tasks in particle filtering are to continuously infer weights $w_{i1}^1, \ldots, w_{i1}^p$ and to select the optimal number of particles p, keeping in mind a particular scenario and type of data [4].

i	t_i	s_j	x_j	y_j	p	v_{ij}^p	w^p
1	01:00	1	3.4	7.2	1	1.1	0.1
1	01:00	1	3.4	7.2	2	3.0	0.6
1	01:00	1	3.4	7.2	3	5.2	0.3
2	01:05	2	5.2	8.5	1	3.1	0.4
2	01:05	2	5.2	8.5	2	7.9	0.3
2	01:05	2	5.2	8.5	3	6.4	0.3

pf_sensor_values

Figure 2.10. Particle filtering stores p weighted sensor values for each time instance t_i.

```
SELECT AVG(v_i1) FROM pf_sensor_values WHERE t > t_start AND t < t_end
```

Query 2.7: Compute the average temperature between time t_{start} and t_{end}.

4.4.2 Static Probabilistic Models. Cheng *et al.* [9–11] model the sensor value as obtained from an user-defined uncertainty range. For example, if the value of a temperature sensor is 15°C, then the actual value could vary between 13°C and 17°C. Furthermore, the assumption is that the sensor value is drawn from a static probability distribution that has support over the uncertainty range.

Thus, for each sensor s_j we associate an uncertainty range between l_{ij} and u_{ij}, in which the actual sensor values can be found. In addition, the pdf of the sensor values of sensor s_j is denoted as $p_{ij}(v)$. Note that the pdf has non-zero support only between l_{ij} and u_{ij}. Consider a query that requests the average temperature of the sensors s_1 and s_2 at time t_i. Since the values of the sensors s_1 and s_2 are uncertain in nature, the response to this query is a pdf, denoted as $p_{avg}(v)$. This pdf gives us the probability of the sensor value v being the average. $p_{avg}(v)$ is computed using the following formula:

$$p_{avg}(v) = \int_{max(l_{i1,v-u_{i2}})}^{min(u_{i1,v-l_{i2}})} p_{i1}(y)p_{i2}(v-x)dx. \tag{2.12}$$

Naturally, Eq. (2.12) becomes more complicated when there are many (and not only two) sensors involved in the query. Additional details about handling such scenarios can be found in [9].

4.5 Query Processing over Semantic States

The MIST framework [5] proposes to use Hidden Markov Models (HMMs) for deriving semantic meaning from the sensor values. HMMs allow us to capture the hidden states, which are sometimes of more interest than the actual sensor values. Consider, as an example, a scenario where the sensors **S** are used to monitor the temperature in all the rooms of a building. Generally, we are only interested to know which rooms are hot or cold, rather than the actual temperature in those rooms. We, then, can use a two-state HMM with states *Hot* (denoted as H) and *Cold* (denoted as C) to continuously infer the semantic states of the temperature in all the rooms.

Furthermore, MIST proposes an in-network index structure for indexing the HMMs. This index can be used for improving the performance of query processing. For instance, if we are interested in finding

the rooms that are *Hot* with probability greater than 0.9, then the in-network model index can efficiently prune the rooms that are surely not a part of the query response. Due to the lack of space, we shall not cover the details of index construction and pruning. We encourage the interested reader to read the following paper [5].

4.6 Processing Event Queries

Event queries are another important class of queries that are proposed in the literature. These queries continuously monitor for a particular event that could probably occur in sensor data. Consider a setup consisting of RFID sensors in a building. An event query could monitor an event of a person entering a room or taking coffee, etc. Moreover, event queries can also be registered, not only to monitor a single event, but a sequence of events that are important to the user. Again, due to space constraints, we shall not cover any of the event query processing approaches in detail. The interested reader is referred to the prior works on this subject [55, 65, 68, 45].

5. Model-Based Sensor Data Compression

Recent advances in sensor technology has resulted in the availability of a multitude of (often privately-held) sensors. Embedded sensing functionality (e.g., sound, accelerometer, temperature, GPS, RFID, etc.) is now included in mobile devices, like, phones, cars, or buses. The large number of these devices and the huge volume of raw monitored data pose new challenges for sustainable storage and efficient retrieval of the sensor data streams. To this end, a multitude of model-based regression, transformation and filtering techniques have been proposed for approximation of sensor data streams. This section categorizes and reviews the most important model-based approaches towards compression of sensor data. These models often exploit spatio-temporal correlations within data streams to compress the data within a certain error norm; this is also known as *lossy compression*. Moreover, several standard orthogonal transformation methods (like, Fourier or wavelet transform) reduce the amount of storage space required by reducing the dimensionality of data.

Unlike the assumptions of Section 2, where we assumed a sensor network consisting of several sensors, here we assume that we only have a single sensor. We have dropped the several sensors assumption to simplify the notation and discussion in this section. Furthermore, we assume that the sensor values from the single sensor are in a form of a *data stream*. Let us denote such a data stream as a sequence of data tuples (t_i, v_i), where v_i is the sensor value at time t_i.

5.1 Overview of Sensor Data Compression System

The goal of the sensor data compression system is to approximate a sensor data stream by a set of functions. Data compression methods that we are going to study in this section permit the occurrence of approximation errors. These errors are characterized by a specific error norm. Furthermore, a standard approach to sensor data compression is to segment the data stream into *data segments*, and then approximate each data segment, so that a specific error norm is satisfied. For example, if we are considering the L_∞ norm, then each sensor value of the data stream is approximated within an error bound ϵ.

Let us assume that we have K segments of a data stream. We denote these segments as $g^1, g^2, \ldots, g^K$, where g^1 approximates the data tuples $((t_1, v_1), \ldots, (t_{i_1}, v_{i_1}))$, while g^k, where $k = 2, \ldots, K$, approximates the data items $((t_{i_{k-1}+1}, v_{i_{k-1}+1}), (t_{i_{k-1}+2}, v_{i_{k-1}+1}), \ldots, (t_{i_k}, v_{i_k}))$. Similar to [20], we distinguish between two classes of the segments used for approximation, namely *connected segments* and *disconnected segments*. In connected segments, the ending point of the previous segment is the starting point of the new segment. On the contrary, in disconnected segments, the approximation of the new segment starts from the subsequent data item in the stream. Disconnected segments offer more approximation flexibility and may lead to fewer segments; however, for linear approximation [35], they necessitate the storage of two data tuples (i.e., start tuple and end tuple) per data segment, as opposed to connected segments.

Since functions are employed for approximating data segments, only the approximated data segments are stored in the database, instead of the raw sensor values of the data stream [64, 50]. A schema for linear segments is presented in [64], consisting of a table, referred to as `FunctionTable`, where each row represents a linear model with attributes `start_time`, `end_time`, `slope` and `intercept` (i.e., base) of the segment. In case of connected segments [20], the `end_time` attribute can be omitted.

A more generic schema for storing data streams, approximated by multiple models, was proposed in [50]. It consists of one table, referred to as the (`SegmentTable`) for storing data segments, and a second table (`ModelTable`) for storing the model functions, as depicted in Figure 2.11. A tuple of `SegmentTable` contains the approximation data for a segment in the time interval `[start_time, end_time]`. The attribute `id` stands for identification of the model that is used in the segment. The primary key in the `SegmentTable` is the `start_time`, while in the

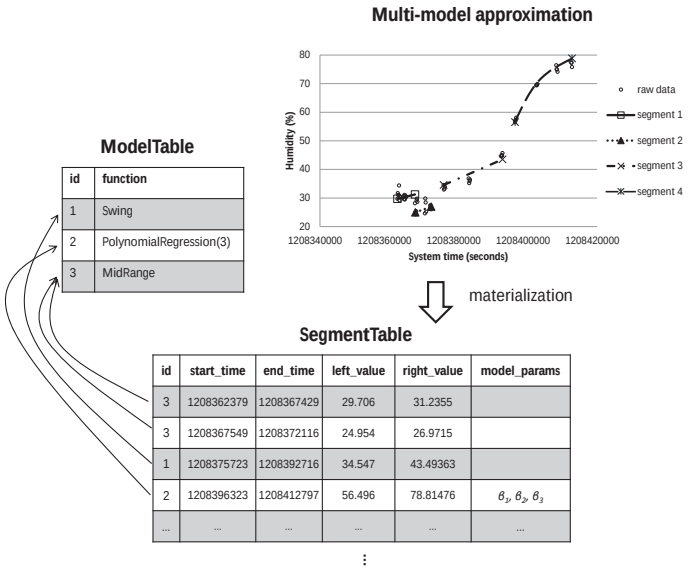


Figure 2.11. The database schema for multi-model materialization.

`ModelTable` it is `id`. When, both, linear and non-linear models are employed for approximation, `left_value` is the lowest raw sensor value encountered in the segment, and `right_value` is the highest raw sensor value encountered in the segment. In this case, `start_time`, `end_time`, `left_value` and `right_value` define a rectangular bucket that contains the values of the segment.

The attribute `model_params` stores the parameters of the model associated with the model identifier `id`. For example, regression coefficients are stored for the regression model. The attribute `model_params` has variable length (e.g., `VARCHAR` or `VARBINARY` data types in SQL) and it stores the concatenation of the parameters or their compressed representation, by means of standard lossless compression techniques (refer Section 5.7) or by a bitmap coding of approximate values, as proposed in [3]. Each tuple in the ModelTable corresponds to a model with a particular `id` and `function`. The attribute `function` represents the name of the model and it maps to the names of two user defined functions (UDFs) stored in the database. The first function implements the mathematical formula of the model, and the second function implements the inverse mathematical formula of the model, if any. Both the UDFs are employed for answering value-based queries. While the first function is used for value regeneration over fixed time steps (also referred to as *gridding*), the second function is used for solving equations.

5.2 Methods for Data Segmentation

In [34], the piecewise linear approximation algorithms are categorized in three groups: sliding window, top-down and bottom-up. The sliding window approach expands the data segment as long as the data tuples fit. The bottom-up approach first applies basic data segmentation employing the sliding window approach. Then, for two consecutive segments, it calculates merging cost in terms of an approximation error. Subsequently, it merges the segments with the minimum cost within the maximum allowed approximation error, and updates the merging costs of the updated segments. The process ends when no further merging can be done without violating the maximum approximation error. The top-down approach recursively splits the stream into two segments, so as to obtain longest segments with the lowest error until all segments are approximated within the maximum allowed error.

Among these three groups, only the sliding window approach can be used online, but it employs look-ahead. The other two approaches perform better than the sliding window approach, but they need to scan all data, hence they cannot be used for approximating streaming data. Based on this observation, Keogh *et al.* [34] propose a new algorithm that combines the online processing property of the sliding window approach and the performance of the bottom-up approach. This approach needs a predefined buffer length. If the buffer is small, then it may produce many small data segments; if the buffer is large, then there is a delay in returning the approximated data segment. The maximum look-ahead size is constrained by the maximum allowed delay between data production and data reporting or data archiving.

5.3 Piecewise Approximation

Among several different data stream approximation techniques, piecewise linear approximation has been the most widely used [34, 39]. Piecewise linear approximation models the data stream with a separate linear function per data segment. Piecewise constant approximation (PCA) approximates a data segment with a constant value, which can be the first value of the segment (referred to as the cache filter) [47], the mean value or the median value (referred to as poor man's compression - midrange (PMC-MR) [39]).

In the cache filter, for all the sensor values in a segment g^k, the following condition should be satisfied:

$$\left|v_{i_{k-1}+p} - v_{i_{k-1}+1}\right| < \epsilon \qquad \text{for } p = 1, \ldots, i_k, \tag{2.13}$$

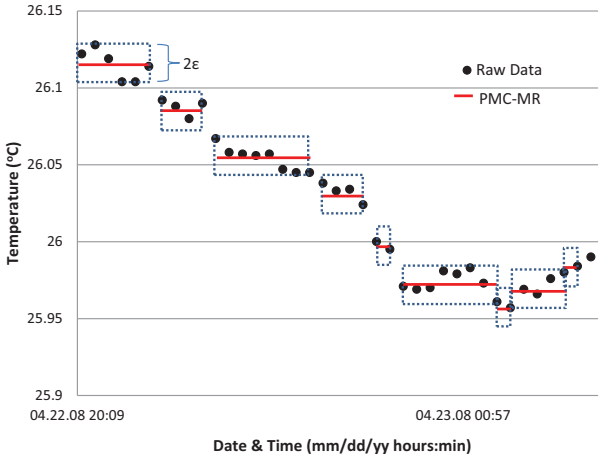


Figure 2.12. Poor Man's Compression - MidRange (PMC-MR).

where ϵ is the maximum allowed approximation error according to the L_∞ norm. Also, for PMC-Mean and PMC-MR the sensor values in a segment g^k should satisfy the following condition:

$$\max_{1 \leq p \leq i_k} v_{i_{k-1}+p} - \min_{1 \leq p \leq i_k} v_{i_{k-1}+p} \leq 2\epsilon \,. \tag{2.14}$$

Furthermore, for PMC-Mean, the approximation value for the segment g^k is given by the mean value of the sensor values in segment g^k. But, for PMC-MR it is given as follows:

$$\frac{\max_{1 \leq p \leq i_k} v_{i_{k-1}+p} - \min_{1 \leq p \leq i_k} v_{i_{k-1}+p}}{2} \,.$$

The data segmentation approach for PMC-MR is illustrated in Figure 2.12.

Moreover, the linear filter [34] is a simple piecewise linear approximation technique in which the sensor values are approximated by a line connecting the first and second point of the segment. When a new data tuple cannot be approximated by this line with the specified error bound, a new segment is started. In [20], two new piecewise linear approximation models were proposed, namely *Swing* and *Slide*, that achieve much higher compression compared to the cache and linear filters. We briefly discuss the swing and slide filters below.

5.3.1 Swing and Slide Filters. The swing filter is capable of approximating multi-dimensional data. But, for simplicity, we describe

its algorithm for one-dimensional data. Given the arrival of two data tuples (t_1, v_1) and (t_2, v_2) of the first segment of the data stream, the swing filter maintains a set of lines, bounded by an upper line u^1 and a lower line l^1. u^1 is defined by the pair of points (t_1, v_1) and $(t_2, v_2 + \epsilon)$, while l^1 is defined by the pair of points (t_1, v_1) and $(t_2, v_2 - \epsilon)$, where ϵ is the maximum approximation error bound. Any line segment between u^1 and l^1 can represent the first two data tuples. When (t_3, v_3) arrives, first it is checked whether it falls within the lines l^1, u^1. Then, in order to maintain the invariant that all lines within the set can represent all data tuples so far, l^1 (respectively u^1) may have to be adjusted to the higher-slope (respectively lower-slope) line defined by the pair of data tuples $((t_1, v_1), (t_3, v_3 - \epsilon))$ (respectively $((t_1, v_1), (t_3, v_3 + \epsilon))$). Lines below this new l^1 or above this new u^1 cannot represent the data tuple (t_3, v_3). The segment estimation continues until the new data tuple falls out of the upper and lower lines for a segment. The generated line segment for the completed filtering interval is chosen so as to minimize the mean square error for the data tuples observed in that interval. As opposed to the slide filter (described below), in the swing filter the new data segment starts from the end point of the previous data segment.

In the slide filter, the operation is similar to the swing filter, but upper and lower lines u and l are defined differently. Specifically, after (t_1, v_1) and (t_2, v_2) arrive, u^1 is defined by the pair of data tuples $(t_1, v_1 - \epsilon)$ and $(t_2, v_2 + \epsilon)$, while l^1 is defined by $(t_1, v_1 + \epsilon)$ and $(t_2, v_2 - \epsilon)$. After the arrival of (t_3, v_3), l^1 (respectively u^1) may need to be adjusted to the higher-slope (respectively lower-slope) line defined by $((t_j, v_j + \epsilon), (t_3, v_3 - \epsilon))$ (respectively $((t_i, v_i - \epsilon), (t_3, v_3 + \epsilon))$), where $i \in [1, 2]$. The slide filter also includes a look-ahead of one segment, in order to produce connected segments instead of disconnected segments, when possible.

Palpanas *et al.* [48] employ *amnesic functions* and propose novel techniques that are applicable to a wide range of user-defined approximating functions. According to amnesic functions, recent data is approximated with higher accuracy, while higher error can be tolerated for older data. Yi and Faloutsos [70] suggested approximating a data stream by dividing it into equal-length segments and recording the mean value of the sensor values that fall within the segment (referred to as segmented means or as piecewise aggregate approximation (PAA)). On the other hand, adaptive piecewise constant approximation (APCA) [6] allows segments to have arbitrary lengths.

5.3.2 Piecewise Linear Approximation. The piecewise linear approximation uses the linear regression model for compressing

data streams. The linear regression model of a data segment is given as:

$$v_i = s \cdot t_i + b, \tag{2.15}$$

where b and s are known as the base and the slope respectively. The difference between v_i and t_i is known as the residual for time t_i. For fitting a linear regression model of Eq. (2.15) to the sensor values $v_i : t_i \in [t_b; t_e]$, the ordinary least squares (OLS) estimator is employed. The OLS estimator selects b and s such that they minimize the following sum of squared residuals:

$$RSS(b, s) = \sum_{t_i=t_b}^{t_e} [v_i - (s \cdot t_i + b)]^2.$$

Therefore, b and s are given as:

$$\begin{aligned} b &= \sum_{t_i=t_b}^{t_e} \left(\frac{t_i - \frac{t_b+t_e}{2}}{\sum_{t_i=t_b}^{t_e} (t_i - \frac{t_b+t_e}{2}) t_i} \right) v_i, \\ s &= \frac{\sum_{t_i=t_b}^{t_e} v_i}{t_e - t_b + 1} - b \frac{t_b + t_e}{2}. \end{aligned} \tag{2.16}$$

Here, the storage record of each data segment of the data stream consists of $([t_b; t_e]; b, s)$, where $[t_b; t_e]$ is the segment interval, and s and b are the slope and base of the linear regression, as obtained from Eq. (2.16).

Similarly, instead of the linear regression model, a polynomial regression model (refer Eq. (2.9)) can also be utilized for approximating each segment of the data stream. The storage record of the polynomial regression model is similar to the linear regression model. The only difference is that for the polynomial regression model the storage record contains parameters $\alpha_1, \ldots, \alpha_d$ instead of the parameters b and s.

5.4 Compressing Correlated Data Streams

Several approaches [14, 42, 24] exploit correlations among different data streams for compression. The GAMPS approach [24] dynamically identifies and exploits correlations among different data segments and then jointly compresses them within an error bound employing a polynomial-time approximation algorithm. In the first phase, data segments are individually approximated based on piecewise constant approximation (specifically the PMC-Mean described in Section 5.3). In the second phase, each data segment is approximated by a ratio with respect to a base segment. The segment formed by the ratios is called the ratio segment. GAMPS proposes to store the base segment and the

ratio segment, instead of storing the original data segment. The idea here is that, in practice, the ratio segment is flat and therefore can be significantly compressed as compared to the original data segment.

Furthermore, the objective of the GAMPS approach is to identify a set of base segments, and associate every data segment with a base segment, such that the ratio segment can be used for reconstructing the data segment within a L_∞ error bound. The problem of identification of the base segments is posed as a *facility location* problem. Since this problem is NP-hard, a polynomial-time approximation algorithm is used for solving it, and producing the base segments and the assignment between the base segments and data segments.

Prior to GAMPS, Deligiannakis *et al.* [14] proposed the self-based regression (SBR) algorithm that also finds a base-signal for compressing historical sensor data based on spatial correlations among different data streams. The base-signal for each segment captures the prominent features of the other signals, and SBR finds piecewise correlations (based on linear regression) to the base-signal. Lin *et al.* [42] proposed an algorithm, referred to as adaptive linear vector quantization (ALVQ), which improves SBR in two ways: (i) it increases the precision of compression, and (ii) it reduces the bandwidth consumption by compressing the update of the base signal.

5.5 Multi-Model Data Compression

The potential burstiness of the data streams and the error introduced by the sensors often result in limited effectiveness of a single model for approximating a data stream within the prescribed error bound. Acknowledging this, Lazaridis *et al.* [39] argue that a global approximation model may not be the best approach and mention the potential need for using multiple models. In [40], it is also recognized that different approximation models are more appropriate for data streams of different statistical properties. The approach in [40] aims to find the best model approximating the data stream based on the overall *hit ratio* (i.e., the ratio of the number of data tuples fitting the model to the total number of data tuples).

Papaioannou *et al.* [50] aim to effectively find the best combination of different models for approximating various segments of the stream regardless of the error norm. They argue that the selection of the most efficient model depends on the characteristics of the data stream, namely rate, burstiness, data range, etc., which cannot be always known *a priori* for sensors and they can even be dynamic. Their approach dynamically adapts to the properties of the data stream and approximates each data

segment with the most suitable model. They propose a greedy approach in which they employ multiple models for each segment of the data stream and store the model that achieves the highest compression ratio for the segment. They experimentally proved that their multi-model approximation approach always produces fewer or equal data segments than those of the best individual model. Their approach could also be used to exploit spatial correlations among different attributes from the same location, e.g., humidity and temperature from the same stationary sensor.

5.6 Orthogonal Transformations

The main application of the orthogonal transformation approaches has been in dimensionality reduction, since reducing the dimensionality improves performance of indexing techniques for similarity search in large collections of data streams. Typically, sequences of fixed length are mapped to points in an N-dimensional Euclidean space; then, multidimensional access methods, such as R-tree family, can be used for fast access of those points. Since, sequences are usually long, a straightforward application of the above approach, which does not use dimensionality reduction, suffers from performance degradation due to the curse of dimensionality [56].

The process of dimensionality reduction can be described as follows. The original data stream or signal is a finite sequence of real values or coefficients, recorded over time. This signal is transformed (using a specific transformation function) into a signal in a transformed space. To achieve dimensionality reduction, a subset of the coefficients of the orthogonal transformation are selected as features. These features form a feature space, which is simply a projection of the transformed space. The basic idea is to approximate the original data stream with a few coefficients of the orthogonal transformation; thus reducing the dimensionality of the data stream.

5.6.1 Discrete Fourier Transform (DFT). The Fourier transform is the most popular orthogonal transformation. It is based on the simple observation that every signal can be represented by a superposition of sine and cosine functions. The discrete Fourier transform (DFT) and discrete cosine transform (DCT) are efficient forms of the Fourier transform often used in applications. The DFT is the most popular orthogonal transformation and was first used in [1, 22]. The Discrete Fourier Transform of a time sequence $x = x_0, \ldots, x_{N-1}$ is a sequence

$X = X_0, \ldots, X_{N-1}$ of complex numbers given by:

$$X_k = \sum_{j=0}^{N-1} e^{-i2\pi \frac{k}{N} j}. \tag{2.17}$$

The original signal can be reconstructed by the inverse Fourier transform of X, which is given by:

$$x_j = \sum_{k=0}^{N-1} X_k e^{i2\pi \frac{k}{N} j}. \tag{2.18}$$

In [1], Agrawal *et al.* suggest using the DFT for dimensionality reduction of long observation sequences. They argue that most real signals only require a few DFT coefficients for their approximation. Thus similarity search can be performed only over the first few DFT coefficients, instead of the full observation sequence. This provides an efficient and approximate solution to the problem of similarity search in high-dimensional spaces. They use the Euclidean distance as the dissimilarity measure.

5.6.2 Discrete Wavelet Transform. Wavelets can be thought of as a generalization of the Fourier transform to a much larger family of functions than sine and cosine. Mathematically, a wavelet is a function $\psi_{j,k}$ defined on the real numbers $\mathbb{R}$, which includes an integer translation by k, also called a shift, and a dyadic dilation (a product by the powers of two), known as stretching. The functions $\psi_{j,k}$ play a similar role as the exponential functions in the Fourier transform: $\psi_{j,k}$ form an orthonormal basis for the $L^2(\mathbb{R})$ space. The $L^2(\mathbb{R})$ space consists of all the functions whose L_2 norm is finite. Particularly, the functions $\psi_{j,k}$, where j and k are integers are given as follows:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k). \tag{2.19}$$

Similar to the Fourier transform, by using the orthonormal basis functions $\psi_{j,k}$, we can uniquely express a function $f \in L^2(\mathbb{R})$ as a linear combination of the basis functions $\psi_{j,k}$ as follows:

$$f = \sum_{j,k \in \mathbb{Z}} < f, \psi_{j,k} > \psi_{j,k}, \tag{2.20}$$

where $< f, g > := \int_R f\overline{g}dx$ is the usual inner product of two functions in $L^2(\mathbb{R})$.

The Haar wavelets are the most elementary example of wavelets. The mother wavelet ψ for the Haar wavelets is the following function:

$$\psi_{\text{Haar}}(t) = \begin{cases} 1, & \text{if } 0 < t < 0.5, \\ -1, & \text{if } 0.5 < t < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{2.21}$$

Ganesan *et al.* [26, 25] proposed in-network storage of wavelet-based summaries of sensor data. Recently, discrete wavelet transform (DWT) was also proposed in [53, 7] for sensor data compression. For sustainable storage and querying, they propose progressive aging of summaries and load sharing techniques.

5.6.3 Discussion. The basis functions of some wavelet transforms are non-zero only on a finite interval. Therefore, wavelets may be only able to capture local (time dependent) properties of the data, as opposed to Fourier transforms, which can capture global properties. The computational efficiency of the wavelet transforms is higher than the Fast Fourier transform (FFT). However, while the Fourier transform can accurately approximate arbitrary signals, the Haar wavelet is not likely to approximate a smooth function using few features.

The wavelet transform representation is intrinsically coupled with approximating sequences whose length is a power of two. Using wavelets with sequences that have other lengths require ad-hoc measures that reduce the fidelity of the approximation, and increase the complexity of the implementation. DFT and DCT have been successfully adapted to incremental computation [72]. However, as each DFT/DCT coefficient makes a global contribution to the entire data stream, assigning less significance to the past data is not obvious with these transformations.

5.7 Lossless vs. Lossy Compression

While lossless compression is able to accurately reconstruct the original data, lossy compression techniques approximate data streams within a certain error bound. Most lossless compression schemes perform two steps in sequence: the first step generates a statistical model for the input data, and the second step uses this model to map input data to bit sequences. These bit sequences are mapped in such a way that frequently encountered data will produce shorter output than infrequent data. General-purpose compression schemes include DEFLATE (employed by gzip, ZIP, PNG, etc.), LZW (employed by GIF, compress, etc.), LZMA (employed by 7zip). The primary encoding algorithms used to produce bit sequences are Huffman coding (also used by DEFLATE)

and arithmetic coding. Arithmetic coding achieves compression rates close to the best possible, for a particular statistical model, which is given by the information entropy. On the other hand, Huffman compression is simpler and faster but produces poor results.

Lossless compression techniques, however, are not adequate for a number of reasons: (a) as experimentally found in [39], gzip lossless compression achieves poor compression (50%) compared to lossy techniques, (b) lossless compression and decompression are usually more computationally intensive than lossy techniques, and (c) indexing cannot be employed for archived data with lossless compression.

6. Summary

In this chapter, we presented a comprehensive overview of the various aspects of model-based sensor data acquisition and management. Primarily, the objectives of the model-based techniques are efficient data acquisition, handling missing data, outlier detection, data compression, data aggregation and summarization. We started with acquisition techniques like TinyDB [45], Ken [12], PRESTO [41]. In particular, we focused on how acqusitional queries are disseminated in the sensor network using routing trees [44]. Then we surveyed various approaches for sensor data cleaning, including polynomial-based [73], probabilistic [21, 63, 52, 65] and declarative [31, 46].

For processing spatial, temporal and threshold queries, we detailed query processing approaches like MauveDB [18], FunctionDB [64], particle filtering [33], MIST [5], etc. Here, our primary objective was to demonstrate how model-based techniques are used for improving various aspects of query processing over sensor data. Lastly, we discussed data compression techniques, like, linear approximation [34, 39, 48], multimodel approximations [39, 40, 50] and orthogonal transformations [1, 22, 53, 7].

All the methods that we presented in this chapter were model-based. They utilized models – statistical or otherwise – for describing, simplifying or abstracting various components of sensor data acquisition and management.

Acknowledgments

This work was supported by the OpenSense project (Nano-Tera reference number 839_401), NCCR-MICS (http://www.mics.org), and by the OpenIoT project (EU FP7-ICT 287305).

References

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[2] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568, 2009.

[3] A. Arion, H. Jeung, and K. Aberer. Efficiently maintaining distributed model-based views on real-time data streams. In *GLOBECOM*, pages 1–6, 2011.

[4] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[5] A. Bhattacharya, A. Meka, and A. Singh. MIST: Distributed indexing and querying in sensor networks using statistical models. In *VLDB*, pages 854–865, 2007.

[6] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*, 27(2):188–228, 2002.

[7] K. Chan and W. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[9] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, pages 551–562, 2003.

[10] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluation of probabilistic queries over imprecise data in constantly-evolving environments. *Information Systems*, 32(1):104–130, 2007.

[11] R. Cheng, S. Singh, and S. Prabhakar. U-DBMS: A database system for managing constantly-evolving data. In *VLDB*, pages 1271–1274, 2005.

[12] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *ICDE*, pages 48–48, 2006.

[13] F. Chu, Y. Wang, S. Parker, and C. Zaniolo. Data cleaning using belief propagation. In *IQIS*, pages 99–104, 2005.

[14] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD*, pages 527–538, 2004.

[15] A. Deligiannakis, V. Stoumpos, Y. Kotidis, V. Vassalos, and A. Delis. Outlier-aware data aggregation in sensor networks. In *ICDE*, pages 1448–1450, 2008.

[16] A. Deshpande, C. Guestrin, W. Hong, and S. Madden. Exploiting correlated attributes in acquisitional query processing. In *ICDE*, pages 143–154, 2005.

[17] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[18] A. Deshpande and S. Madden. MauveDB: Supporting model-based user views in database systems. In *SIGMOD*, pages 73–84, 2006.

[19] R. Elmasri and S. Navathe. *Fundamentals of database systems*. Addison Wesley, 6$^{\text{th}}$ edition, 2010.

[20] H. Elmeleegy, A. Elmagarmid, E. Cecchet, W. Aref, and W. Zwaenepoel. Online piece-wise linear approximation of numerical streams with precision guarantees. In *VLDB*, pages 145–156, 2009.

[21] E. Elnahrawy and B. Nath. Cleaning and querying noisy sensors. In *WSNA*, pages 78–87, 2003.

[22] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[23] C. Franke and M. Gertz. ORDEN: Outlier region detection and exploration in sensor networks. In *SIGMOD*, pages 1075–1077, 2009.

[24] S. Gandhi, S. Nath, S. Suri, and J. Liu. GAMPS: Compressing multi sensor data by grouping and amplitude scaling. In *SIGMOD*, pages 771–784, 2009.

[25] D. Ganesan, D. Estrin, and J. Heidemann. DIMENSIONS: Why do we need a new data handling architecture for sensor networks? In *SIGCOMM*, pages 143–148, 2003.

[26] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and H. J. An evaluation of multi-resolution storage for sensor networks. In *SenSys*, pages 89–102, 2003.

[27] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: An efficient framework for modeling sensor network data. In *IPSN*, pages 1–10, 2004.

[28] H. Gupta, V. Navda, S. Das, and V. Chowdhary. Efficient gathering of correlated data in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(1):4, 2008.

[29] A. Jain, E. Chang, and Y.-F. Wang. Adaptive stream resource management using Kalman Filters. In *SIGMOD*, pages 11–22, 2004.

[30] S. Jeffery, G. Alonso, M. Franklin, W. Hong, and J. Widom. A pipelined framework for online cleaning of sensor data streams. In *ICDE*, page 140, 2006.

[31] S. Jeffery, G. Alonso, M. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. In *Pervasive*, pages 83–100, 2006.

[32] S. Jeffery, M. Garofalakis, and M. Franklin. Adaptive cleaning for RFID data streams. In *VLDB*, pages 163–174, 2006.

[33] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, pages 1160–1169, 2008.

[34] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.

[35] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *SIGKDD*, pages 239–241, 1998.

[36] A. Klein. Incorporating quality aspects in sensor data streams. In *PIKM*, pages 77–84, 2007.

[37] A. Klein and W. Lehner. Representing data quality in sensor data streaming environments. *Journal of Data and Information Quality*, 1(2):1–28, 2009.

[38] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *ICDE*, pages 131–142, 2005.

[39] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *ICDE*, pages 429–440, March 2003.

[40] Y. Le Borgne, S. Santini, and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing*, 87(12):3010–3020, 2007.

[41] M. Li, D. Ganesan, and P. Shenoy. PRESTO: Feedback-driven data management in sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 17(4):1256–1269, 2009.

[42] S. Lin, V. Kalogeraki, D. Gunopulos, and S. Lonardi. Online information compression in sensor networks. In *IEEE International Conference on Communications*, 2006.

[43] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[44] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD*, pages 491–502, 2003.

[45] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TinyDB: An acquisitional query processing system for sensor networks. *TODS*, 30(1):122–173, 2005.

[46] C. Mayfield, J. Neville, and S. Prabhakar. ERACER: A database approach for statistical inference and data cleaning. In *SIGMOD*, pages 75–86, 2010.

[47] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*, pages 563–574, 2003.

[48] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel. Online amnesic approximation of streaming time series. In *ICDE*, pages 339–349, 2004.

[49] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[50] T. Papaioannou, M. Riahi, and K. Aberer. Towards online multi-model approximation of time series. In *IEEE MDM*, pages 33–38, 2011.

[51] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, 2007.

[52] A. Petrosino and A. Staiano. A neuro-fuzzy approach for sensor network data cleaning. In *KES*, pages 140–147, 2007.

[53] I. Popivanov. Similarity search over time series data using wavelets. In *ICDE*, pages 212–221, 2002.

[54] J. Rao, S. Doraiswamy, H. Thakkar, and L. Colby. A deferred cleansing method for RFID data analytics. In *VLDB*, pages 175–186, 2006.

[55] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu. Event queries on correlated probabilistic streams. In *SIGMOD*, pages 715–728, 2008.

[56] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[57] S. Sathe, H. Jeung, and K. Aberer. Creating probabilistic databases from imprecise time-series data. In *ICDE*, pages 327–338, 2011.

[58] M. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis. TiNA: A scheme for temporal coherency-aware in-network aggregation. In *MobiDE*, pages 69–76, 2003.

[59] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *MobiHoc*, pages 219–228, 2007.

[60] R. Shumway and D. Stoffer. *Time series analysis and its applications*. Springer-Verlag, New York, 2005.

[61] A. Silberstein, R. Braynard, G. Filpus, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang. Data-driven processing in sensor networks. In *CIDR*, 2007.

[62] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB*, pages 187–198, 2006.

[63] Y. Tan, V. Sehgal, and H. Shahri. SensoClean: Handling noisy and incomplete data in sensor networks using modeling. Technical report, University of Maryland, 2005.

[64] A. Thiagarajan and S. Madden. Querying continuous functions in a database system. In *SIGMOD*, pages 791–804, 2008.

[65] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy. Probabilistic inference over RFID streams in mobile environments. In *ICDE*, pages 1096–1107, 2009.

[66] D. Tulone and S. Madden. PAQ: Time series forecasting for approximate query answering in sensor networks. In *EWSN*, pages 21–37, 2006.

[67] L. Wang and A. Deshpande. Predictive modeling-based data collection in wireless sensor networks. In *EWSN*, pages 34–51, 2008.

[68] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD*, pages 407–418, 2006.

[69] Y. Yao and J. Gehrke. Query processing in sensor networks. In *CIDR*, 2003.

[70] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

[71] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Survey & Tutorials*, 12(2), 2010.

[72] Y. Zhu and D. Shasha. StatStream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.

[73] Y. Zhuang, L. Chen, X. Wang, and X. Lian. A weighted moving average-based approach for cleaning sensor data. In *ICDCS*, page 38, 2007.

Chapter 3

QUERY PROCESSING IN WIRELESS SENSOR NETWORKS

Lixin Wang
Hong Kong University of Science and Technology
leaing@gmail.com

Lei Chen
Hong Kong University of Science and Technology
leichen@cse.ust.hk

Dimitris Papadias
Hong Kong University of Science and Technology
dimitris@cse.ust.hk

Abstract Recently, with the fast development of sensing and wireless communication technology, wireless sensor networks (WSNs) have been applied to monitor the physical world. A WSN consists of a set of sensor nodes, which are small sensing devices with limited computational resources able to communicate with each other located in their radio range. Network protocols ensure the effectiveness of communication between sensor nodes and provide the foundation for WSN applications. The characteristics of WSNs, including the limited energy supply and computational resources, render the design of WSN algorithms challenging and interesting. Both the Database and Network communities have dedicated considerable efforts to make WSNs more effective and efficient. In this chapter, we survey the problems arisen in practical applications of WSNs, focusing on various query processing techniques over captured sensing data.

Keywords: Query Processing, Wireless Sensor Networks

1. Introduction

With the fast development of sensing and wireless communication technology, wireless sensor networks (WSNs) become popular tools to capture the physical worlds. A sensor is a device with one or several sensing devices, a radio component, and limited computational resources. It takes physical measurements of the environment, e.g., temperature, light, sound, and humidity. A wireless sensor network (WSN) consists of a base station and a set of sensor nodes. Each node is able to directly communicate with others within its radio coverage. The base station, also called the data sink, is equipped with a radio component so that it is able to communicate with the nearby sensor nodes and collect their captured data. Sensors far away from the sink will transmit the data to the sensors near the sink first, and then the data are relayed to the sink. Depending on the size of a monitoring area, data captured by the sensors located on the boundary of the monitor area may need to relay multiple hops (sensors) before they reach the sink. To query the sensing data, users submit queries at the base station, which then reports the query results. WSNs were first applied in military and scientific projects. Applications of WSNs flourish as the cost of sensors drop, while the capabilities increase. In the past few years, WSNs have attracted considerable interest from both the Network and Database communities. Assume, for instance, a WSN that monitors the physical status of a forest. An environmentalist is interested in the temperature readings, while a biologist is interested in the level of soil moisture. In order to capture different requirements, they submit queries at the base station. In many applications, the query language is declarative and similar to SQL. Q_1 shows a typical of an extreme value monitoring query [4 5 6], which monitors the maximum temperature readings of the forest.

```
Q1:   SELECT   MAX TEMPARTURE
      FROM     SENSORS
      WHERE    SAMPLE INTERVAL=5 mins
```

Q_1 treats the sensor data as a relational table and the temperature reading as an attribute of this table. The SAMPLE INTERVAL clause specifies the cycle length of this network, i.e., the interval between two data collecting activities. Since sensors are battery-powered, it is crucial to minimize their energy consumption in order to prolong the life time of the network, especially when the sensors are deployed in harsh or difficult-to-access environments, e.g., wildlife tracking [2] and habit monitoring [3]. Making sensor nodes working in cycles is a standard way for energy saving in WSNs. Within a cycle, a sensor

- collects measurements from the environment
- receives data from other sensors in its network neighborhood
- possibly performs some computations on the received and collected data
- broadcasts data to the WSN
- enters the sleep mode until it wakes up for the next cycle

Two major challenges in WSNs refer to the effectiveness of communication between sensor nodes and the efficiency of data processing. The first challenge spawns the design space of network protocols. Various protocols ensure the smooth and automatic communication between sensor nodes [4]. A network protocol specifies when a sensor node samples the environment, to whom it reports to and from whom it receives data. There are three categories of protocols, according to how they organize the network: i) tree-based topologies, ii) multi-path-based topologies and iii) the hybrid ones that combine the first two approaches. The second challenge, i.e., the efficiency of data processing, is mostly the focus of the Database community. Several observations realize this task from different aspects. Firstly, in most applications, only a small part of the data is interested by the user. Using descriptive queries like Q_1 helps drop uninteresting data as soon as possible. Secondly, sensor data often exhibit strong spatial-temporal correlations [40]. It is reasonable to predict the sensor readings with certain confidence, based on historical and easily observable sensor readings. Finally, when imprecise answers are allowed, returning approximated ones (with error bounds) often lead to large energy savings. This survey focuses on query processing for WSNs, mostly from the database point of view. The remainder of this chapter is organized as follows. Section 2 introduces the characteristics and limitations of sensor nodes, which motivate the challenges involved in designing WSN systems. Section 3 surveys common topologies of WSNs; different algorithms optimize certain topologies. Section 4 introduces the data storage techniques in WSNs; some applications require the data generated by sensor nodes to be stored inside the network for a period of time. Section 5 discusses data acquisition and aggregation techniques in WSNs; a topic widely investigated in the Database community. Section 6 studies the model-based data acquisition, probabilistic queries and event detection in WSNs. Finally, Section 7 concludes this chapter.

2. Limitations of Sensor Nodes

A sensor is a small device, whose volume is only a few cubic centimeters [7]. It is capable of sensing the environment, communicating with other sensors, and performing simple computations. Section 2.1 introduces energy constraints on sensors, while Section 2.2 summarizes other restrictions such as unreliable transmission, and limited computational resources.

2.1 Energy Constraint

Sensors are powered by batteries. A sensor is dead once it runs out of battery power. The energy constraint is usually the bottleneck for most WSNs. A sensor node consumes extremely low power during the sleep mode, when most of its components are inactive. However, the use of radio component, sensing unit and CPU is extremely power-consuming [8]. A practical WSN, which is expected to be functioning for months [2 3], requires algorithms that properly use the components of sensors. For instance, a Mica node, powered by two AA batteries, runs out of energy in a few days if its components are constantly active. On the other hand, it achieves six months of lifetime when it is properly programmed [7]. The radio component is the most energy-consuming. It serves the three functions of sending messages, receiving messages, and listening for transmission requests from other nodes.

In order to execute a query that has been submitted to the base station, the network protocol spreads it to the (selected) sensor nodes, which transmit their measures back to the base station. The communication cost of query execution includes the total number of messages transmitted in the network in order to answer this query. In particular, it covers functions i) and ii) listed in the above paragraph. There are various algorithms to reduce the communication cost [9][10][11][12]. However, recent research shows that the time that the radio is active dominates the energy cost, rather than the number of messages transmitted. When the radio is on, the sensor is either sending/receiving messages, or listening for incoming transmissions. Table 3.1 shows the typical power consumptions of the radio in different modes. We observe that message transmitting and listening cost substantial energy. Therefore, to reduce the energy consumption, WSN algorithms should have the radio in sleep mode as long as possible.

Table 3.1. Energy consumption of radio in different status

Radio Status	Power
Sending	60mW
Receiving	45mW
Listening	45mW
Sleeping	90μmW

2.2 Other Constraints

Sensors have limited computational and communication resources. The technology of sensors has evolved for several generations. Mica [37] is a popular series of second-generation commercial sensors. A Mica node has a 4 MHz, 8 bit Atmel microprocessor and a 40 Kbits/second radio device. More advanced sensors, such as Mica2, a third generation commercial sensor, are equipped with 7 MHz processors, 128Kbytes program flash memory and radio devices with bandwidth 38.4 Kbits/sec. Even for Mica2, the computational and communication resources are extremely valuable; i.e., it is undesirable for an algorithm on sensor nodes to store large amounts of data and perform complex computations. During data transmission, there is always an upper bound on the packet size. A large message is divided to fit in the packet before transmission. Similar to other wireless radios, the radio devices of Mica nodes are half-duplex; i.e., they are not able to listen to the incoming signals during message transmission. In order to avoid collisions, before sending a message, the nodes listen and detect whether the transmitting channel is in use. If so, they delay their own transmissions for a random period of time and then try again. In general, unreliability in WSNs is attributed to the following aspects:

- Sensor nodes are unreliable. A sensor may occasionally take wrong sample readings. Errors may occur during message transmission. More severely, a node may stop functioning for a short period of time and come back again.

- The links between sensors are unreliable. Since the links are wireless, they are sensitive to the physical status of the environment. Turbulences in the environment (e.g., some object passing through, the change of humidity, etc.) may affect the link quality.

Thus, an important parameter in a WSN is its packet loss rate, which indicates the average probability that a packet gets lost in the path to the base station. Algorithms developed for wireless sensor networks should be able to handle message losses and possibly recover the missing data.

3. Topologies of WSNS

The connectivity graph is a common tool to represent the connections between sensors of a WSN. A WSN consists of a set of sensors, V, each of which acts as a node of the graph. We hereupon use the terms sensor and node interchangeably. A node $n_i \in V$ connects to another node $n_j \in V$ if n_j is within the radio range of n_i, that is, n_i directly communicates with n_j. For such pair of nodes, we add an edge $e_{i,j}$ from n_i to n_j. The set of sensors along with the connections consist the connectivity graph $G =< V, E >$ of the network [38]. In most applications, the sensors are of the same type and have the same radio range. Thus, the connectivity between n_i and n_j is bidirectional. For simplicity, it is a common practice to treat the connectivity of a WSN as an undirected graph. A naive approach for message transmitting is called flooding. In flooding, whenever a node receives a message, it broadcasts the message to its neighbor sensors (i.e., sensors are within the radio range). However, this simple approach is extremely energy consuming due to the large amount of redundant transmissions. In order to facilitate effective messages transmission, the network protocols organize WSNs into certain topologies. Network protocols for WSNs follow various approaches depending on the desired trade-off between communication overhead and robustness. There are mainly three types of topologies: the tree-based topology, the multi-path-based topology and the hybrid topology. In tree-based topologies [4 13], every pair of nodes communicate through a single path. This minimizes the transmission cost, but is very sensitive to packet loss and node failures, which happen frequently in WSNs. Specifically, when a transmission or a node fails, the data from the corresponding sub-tree are lost. On the other hand, multi-path-based topologies [12 14] allow a message to propagate through multiple paths until it reaches the base station, so that even if it gets lost in one path, it is able to be successfully delivered through another one. The trade-off is the higher communication cost and possibly duplicated results compared to the tree-based approaches. Hybrid approaches [9] organize part of the WSN (e.g., reliable nodes with stable communication links) using a tree-based topology, and the rest according to a multi-path approach. In the following, we discuss the three kinds of topologies in Sections 3.1, 3.2 and 3.3, respectively.

3.1 Tree-Based Topology

Given the connectivity graph G, a tree-based topology [4][13] constructs a spanning tree T of G rooted at the base station. The base station acts as the data sink and the entrance of the network. On re-

ceiving a query, the base station propagates it to its children. This query continues propagating level by level between parent-child links until it reaches the leaf nodes. Suppose the query never affects parts of the network, it is energy-efficient to stop the query from being propagated into leaf levels as early as possible. TinyDB [7] achieves this by building the semantic routing tree, SRT, to guide the query dissemination. SRT is a routing tree embedded with a semantic index, which is an index on each node n_i built according to the sensor readings in the sub-tree rooted at n_i. In particular, the user specifies the queried sensors by restricting their attribute(s), A, e.g., ID, location, etc. Conceptually, SRT is an overlay index upon T and is maintained over time. In order to build SRT, for each attribute a_x in A, a node n_i collects and keeps a_x's range (i.e., an aggregation of a_x readings in the sub-tree rooted at n_i) as the index. When a query arrives, n_i checks whether the ranges kept in the index intersects with the query range. If there is no intersection, the query does not overlap with the range represented by its sub-tree and it is dropped at n_i. On receiving the query, a node senses the environment retrieves the physical readings and performs local computation. The query results are transmitted bottom up on T. If the query asks for an aggregation, e.g., maximum value, sum, average, etc., in-network aggregation is performed in order to reduce the sizes of the transmitted messages [4 13]. The advantage of tree-based topology is energy efficiency, since each node sends messages only to its parent. Figure 3.1(a) gives a tree-based WSN topology, where the black node represents the base station and gray nodes are sensors. The solid and dashed lines connecting nodes are bidirectional physical radio connections. However, the latter are conceptually removed by the routing protocol, i.e., there are no data transmissions on those connections. Several works aim at enhancing the robustness of tree-based topologies. A common approach is to make the routing tree recoverable when some nodes stop functioning. Specifically, each node n_i maintains a table of neighboring nodes and it periodically examines the link quality with its current parent. Once the link is broken, n_i sends requests to its neighbors asking for a new parent and reports to the new parent once the request is accepted. However, algorithms have to consider the possible duplications of the sensor readings during the handing over stage. Another approach to partially overcome the vulnerability of the tree-based topology is to build multiple trees on the wireless sensors. Each piece of data is able to reach the base station through multiple paths. Obviously, this approach sacrifices energy efficiency for robustness.

The routing tree can be optimized according to different criteria [15 16 17], e.g., link quality, energy efficiency, responsiveness, etc. However,

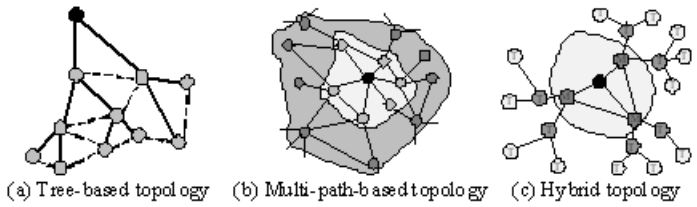


Figure 3.1. Network topologies

most of the data processing techniques do not depend on the specific routing tree. Instead, the resulting network should have the following two properties:

1 It should be able to deliver query requests to all nodes in the network.

2 It should be able to transmit data from every node to the base station.

TinyDB [7], TAG [13], CONCH [10] and HAT [11] are based on the tree-based topologies. In addition, some algorithms require that the routing protocol generates no data duplication during transmission [13].

3.2 Multi-Path-Based Topology

In a multi-path-based topology [12], each node has multiple children and parents. A sensor receives data from its children, processes the data and broadcasts the partial results to all its parents. Consequently, a message reaches the base station through multiple paths, increasing the possibility for data to be successfully delivered. However, if several copies of a reading reach the base station through different paths, duplication arises and the algorithm has to take care of it. A popular multi-path-based topology is Rings [14], where each node works in a certain level. The topology is constructed iteratively starting from the base station n_0, which is in level $l_0 = 0$. To initiate the construction procedure, n_0 broadcasts a signal m_0, which contains the level information of itself, i.e., $m_0.level = 0$. On hearing m_0, a node n_i checks the level l_i of itself. If l_i has not been assigned yet, n_i assigns l_i as $m_0.level + 1$, i.e., one level lower than its parent. Otherwise, l_i is already assigned, n_i sets l_i to $min\{l_i, m_0.level + 1\}$, i.e., to be as close to the base station n_0 as possible. After updating its level information, n_i broadcasts a signal q_i with $q_i.level = l_i$ to its neighborhood. This procedure keeps propagating until the level information of all nodes is stable. As a result, each node

n_i is in a level whose value equals to the minimum number of hops from n_i to n_0. This is called minimum-hop routing [39]. If there is a fixed packet loss rate for each hop, minimum-hop routing minimizes the message losses. Figure 3.1(b) shows an example of the Rings topology. As before, the black node represents the base station while other nodes are sensors. We omit the physical connections and only present the connections maintained in the routing protocol. The sensors are divided into two levels, which are represented by different gray scales. In order for n_i to transmit a message m_i to n_0, it attaches its level information along with m_i, i.e., $m_i.level = l_i$, and broadcasts m_i. A node n_j hears m_i checks whether l_j equals to $m_i.level - 1$. If so, it is one of the parents of n_i and broadcasts m_i after setting $m_i.level$ to l_j; the process is repeated until m_i eventually reaches n_0 through multiple paths.

3.3 Hybrid Topology

In a network with high link quality, trees are preferable to multi-path topologies because of their energy efficiency. On the other hand, if the network suffers low link quality, it is better to use a multi-path-based topology for robustness. Manjhi et al. [9] propose a hybrid topology, called the Tributaries and Deltas, which adjusts the topology in different areas of the network according to the local link qualities. The motivation is to reduce energy consumption in low-packet-loss-rate areas, while increasing robustness in high-packet-loss-rate areas. Figure 3.1(c) shows an example of the Tributaries and Deltas topology. The black node represents the base station while gray ones correspond to sensors. The nodes located in the gray area apply a multi-path-based topology, while the rest form trees. The overall network topology is a directed graph, where the direction of an edge agrees with the direction of the data flow, i.e., from outer nodes towards the base station. The nodes labeled with T (resp. M) run the tree-based (resp. multi-path-based) topology. An edge is assigned with the same label as its source node. Generally, trees incur no duplicate data transmissions, as opposed to multi-path-based topologies. In order to ensure the correctness of the aggregation results, the authors propose two constraints:

- Edge Correctness: An M edge can never be incident on a T vertex, i.e., an M edge is always between two M vertices.
- Path Correctness: In any path in the directed graph, a T edge can never appear after an M edge.

The two constraints are actually equivalent. Either of them ensures that a multi-path partial result can only be received by a node in the

multi-path topology (and hence equipped to handle duplication). An implication from the constraints is that the region running multi-path-topology will finally be a sub-graph of the connectivity graph including the base station. As shown in Figure 3.1(c), the outer regions form trees, while the region around the base station is multi-path-based. The nodes at the boundary of the multi-path region are the switchable M nodes, meaning that they can be switched to T nodes without violating the correctness constraints. Also, the T nodes at the boundary of multi-path region are called switchable T nodes, meaning that they can be switched to M nodes freely. In order to adaptively adjust the topology of the network according to the packet loss rates in different areas, an aggregate on the data loss rate is maintained by each node n_i. Specifically, n_i computes the packet loss rate in its sub-tree. Once the packet loss rate exceeds a user specifies threshold ϵ, the sub-tree rooted at n_i suffers high packet loss rate and applies the multi-path-based topology. Otherwise, tree-based topology is applied. Changing between the two topologies is accomplished by switching certain sensors between T and M nodes, so that the multi-path region expands towards the areas with high packet loss rate, while tree-based regions expand towards areas with low packet loss rate.

4. Data Storage

Some applications do not involve a base station. For instance, often scientists deploy sensors in the wild to monitor the habitat of animals [2][3]. In such applications, the nodes form a WSN, which do not have a base station. In order to collect data, scientists drive a vehicle with a data collecting device through the monitoring territory. During the life time of the network, the nodes store readings until they are contacted by the collector. There are two main challenges for such applications: i) due to the limited storage capacity, sensors memories may overflow and ii) workload varies on different areas of the network, e.g., sensors in the areas with frequent activities generate more data than those in areas with rare activities. These issues raise challenges on how to store data evenly in each node, and how to retrieve relevant data in different parts of the network with low cost. [18] divides WSN storage techniques into two categories: centralized and decentralized. In the centralized storage, data are stored on the node that generates them. As an example, in TinyDB, in order to perform certain kinds of aggregated queries, sensors may store a small set of data locally [7]. This technique is not suitable for an environment with frequent burst activities since they quickly drain the valuable memory resource. A popular decentralized storage approach

is the data-centric storage [19 20 21 22 23 24 25]. In data-centric storage, the location to store a piece of data is determined by a set of attributes of the data. The benefit of such scheme is that all the related data could be stored together. Sophisticated algorithms are needed to determine where a piece of data should be stored so as to balance the storage cost of all nodes. A. Omotayo et al. [18] build a model for a data-centric storage scheme with respect to the energy cost for storing and retrieving data in WSNs. Suppose a piece of data d is generated by node n_{src} and is stored at node n_{dest}. The total cost of storing d contains three components: (i) Reading d from the memory of n_{src}, (ii) transmitting d to n_{dest}, and (iii) writing d to the memory of n_{dest}. The total cost of retrieving d contains three components: (i) Routing the retrieval request to n_{dest}, (ii) reading d from the memory of n_{dest}, and (iii) returning d to the base station.

Suppose a node n_i stores part of its data (with size $M_{i,j}$) at another node n_j. The energy cost of storing and retrieving data increases along with the following parameters: i) the distance of ni from the base station, ii) the size of $M_{i,j}$, iii) the distance between n_i and n_j, and iv) the distance of n_j from the base station. These critical parameters are crucial to reducing the energy cost for data-centric storage.

5. Data Acquisition and Aggregation

This section surveys data acquisition and aggregation techniques. Sections 5.1 and 5.2 introduce query models and general frameworks for data aggregation and acquisition. Section 5.2 surveys efficient algorithms for specific applications. Section 5.3 investigates secure aggregation in WSNs. Section 5.4 discusses an extension of the general frameworks to support efficient in-network joins.

5.1 Query Models

Since sensors acquire samples of the environmental parameters periodically, the data from the WSNs are streams. There are two query models in WSNs: push-based and pull-based. In the push-based model, the user registers a continuous query at the base station n_0. The query is then disseminated by n_0 and stored in the network for a relatively long period of time, during which the sensors continuously generate the results that satisfy the query and push them to the base station. This model is the most common and practical one in WSNs. A typical query over the WSN contains the following information: (i) The sampling frequency: how often the sensors take samples, e.g., once per minute, (ii) the affected attributes: which attributes should be sampled, e.g., tem-

perature, and (iii) constraints on the returned values: filter out undesired values, e.g., temperature readings above $100°C$ should be dropped for an application monitoring the water temperatures.

For the pull-based model, a snap shot result is returned for a query. Specifically, n_0 disseminates a query into the network. On receiving the query, a sensor n_i returns its current reading. After n_0 receives all the responses, it generates and returns the final result at the current time stamp to the user. As an example, a query in the pull-based model is "reporting the current temperature on the node with $ID = 2$". The main difference between these two models is that the push-based one returns a stream of results, while the pull-based one returns only one result which is the snap shot of the current network status.

5.2 Basic Acquisition and Aggregation

In early WSNs, the collected data were transferred to and processed at the base station, regardless of their usefulness. Such systems lack flexibility and scalability [7] because: they take samples in a fixed manner and, they transmit large amounts of raw data. Consequently, they have no control on which attributes to retrieve, the range of the returned readings, etc. Current systems provide control on the sensor's behavior and offer various optimization opportunities. A WSN can be considered as a database that includes two sets of data: sensor meta data and sensor sensing data. sensor meta data refer to information about the sensors, such as the sensors' IDs, locations, and other physical characteristics. Sensor sensing data are measurements collected from the sensors over time. In COUGAR [27 28], the meta data form a relational table $R(sensor\_node, location)$ at the base station, where $sensor\_node$ indicates the ID of a sensor n_i while location records the physical coordinates of n_i. The sensing data are generated by sensors at each time stamp. COUGAR follows the sequence model introduced by Seshadri et al. [26] and embeds each reading with the time stamp when it is generated. Given a set of tuples embedded with time stamps, a time series of the readings is constructed by sorting the records according to the time stamps.

COUGAR includes an SQL-like declarative language. As an example, a query is specified in the following form:

```
Q2:   SELECT   R.sensor.attribute(range)
      FROM     R
      WHERE    condition  AND  $every(period)
```

The "SELECT" clause specifies that the sensors sample the specific attribute and return only those readings falling in range. The "WHERE"

clause constrains the sensors affected by the query. "*every*(*period*)" is a new expression introduced in COUGAR, indicating a continuous query where each sensor should return a sample every period of time. As an example, the query "every minute, return the abnormal temperatures (i.e., greater than $40°C$) measured by nodes in the 4th floor" can be written as follows:

```
Q3:   SELECT   R.sensor.temperature>40
      FROM     R
      WHERE    R.S.floor  AND  $every(1 minute)
```

The first version of COUGAR does not support aggregation queries based on time windows. For instance, the query "return the average temperature on each floor over the last 10 minutes" cannot be translated into the above declarative language. Bonnet et al. [27] enhance COUGAR to support window queries. COUGAR is suitable for both data acquisition and aggregation in WSNs. However, several applications only require aggregations, without the raw data. Madden et al. [13] develop TAG (short for the Tiny AGgregation service) for aggregation in low-power, distributed, wireless environments. Their goal is to retrieve aggregation information from the WSNs with low energy consumption. TAG develops a declarative language for continuous aggregate queries, similar to COUGAR. Madden et al. [7] introduce another sophisticated system called TinyDB. Similar to the above two systems, TinyDB allows user to specify the data collecting manner using a declarative language. It focuses on reducing the energy cost for acquiring and transmitting data. The data generated by sensors form a single conceptual table called sensors. Each kind of measurement, e.g., humidity, temperature or light strength, forms a field in sensors. A tuple contains the samples of different measurements acquired by a sensor at a single time-stamp. Newly acquired tuples are appended at the end of sensor. A query in TinyDB consists of SELECT, FROM, WHERE and GROUPBY clauses. In addition, TinyDB incorporates new key words "SAMPLE PERIOD" and "FOR" to specify the frequency of taking samples and the life time of the query, respectively. An example of a typical query is as follows:

```
Q4:   SELECT   nodeid,light, temperature
      FROM     sensors
      WHERE    SAMPLE PERIOD  $every(1 s)
```

This query requires the sensors to take samples of light and temperature every 1 second for a period of 10 seconds. Each time a set of samples is acquired, the sensor should return it to the base station together with its node id. The results are streams, one for each sensor, lasting for 10 seconds. They will be finally forwarded to the base station

via a multi-hop topology. Each tuple of each stream includes a time stamp corresponding to the time it was produced. In some applications, the lifetime of the network is much more important than the sampling frequency. For example, in a wild life habitat monitoring application, scientists may not be aware of how the sampling frequency will affect the life time of the network. However, they want the network work at least one month. The keyword "LIFETIME" is introduced in TinyDB to specify at least how long the network should function:

```
Q5:   SELECT     nodeid,movements
      FROM       sensors
      LIFETIME   30 days
```

In this case, TinyDB performs life time estimation to adjust the sampling frequency (at the same time, the frequency of sending and receiving messages), so that the remaining energy can last until the specified life time. According to the energy costs of accessing sensors, the selectivity of the query, the expected communication rates and the remaining energy, a sampling frequency is computed to ensure the expected life time.

Silberstein et al. [10] propose CONstrain CHaining (CONCH) for data acquisition. The main idea of their approach is to provide effective spatio-temporal suppression and use a minimum spanning forest of the network for data transmission. The CONCH method is an edge monitoring approach that exploits spatial-temporal correlations in sensor readings and effectively reduces the message transmission based on such correlation. As data transmission dominants the energy consumption, CONCH exhibits considerable energy reduction during the empirical evaluation. CONCH is based on the tree topology. The base station directly monitors the readings of a selected portion of sensors, which report their readings to the base station. For a remaining node n_i, it reports to its parent, which computes the difference on the reading between itself and n_i. The parent then reports to the base station if and only if the difference changes. The base station monitors such differences on chains of nodes. It assumes the difference between the corresponding pair of nodes does not change if no report is received. Knowing the global topology of the network, the base station is able to recover each node's reading from the readings of the directly monitored nodes and the chained differences. Due to the spatio-temporal correlation, the readings of nearby sensors always share the same trend and the differences between them do not change frequently. So a large amount of transmissions are suppressed and CONCH has good performance.

5.3 Secure Aggregation

A WSN generates huge amount of data. Computing aggregations on these data may bring significant cost to the owner of the network. Fortunately, there are third-party aggregators, who have the advantage of expertise consolidation, are able to provide aggregation service on the raw data with lower cost and better performance. Figure 3.2 shows the model of outsourced aggregation. The aggregator lies between sensors and the portal, which are both facilities belonging to the data owner. The portal, acting as a proxy, delivers data, i.e., queries and results, between user and the aggregator. Since the aggregator is usually untrusted, the portal has to verify the results. The verification ensures the results are: i) correct: the results are indeed originated from the sensors, not faked by the aggregator. ii) complete: all data belong to the result set are returned, none is dropped.

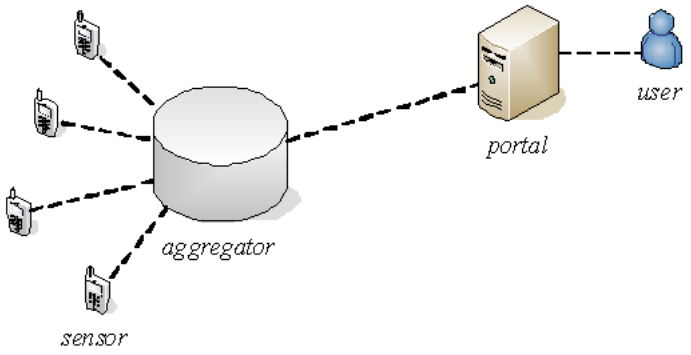


Figure 3.2. Outsourced aggregation model

As a general setting, each sensor n_i shares a secret k_i with the portal, which is unknown to the aggregator. The approach to ensure the correctness is straightforward. For each raw data d, n_i computes a Massage Authentication Code (MAC for short) md using k_i. n_i then sends the pair $< d, m_d >$ to the aggregator. The aggregator cannot generate proper MAC for a message since it is not aware of k_i, while the portal is able to verify the correctness using the shared keys. However, to ensure the completeness is much more challenging. Nath et al. [41] propose SECOA for secure aggregation on the maximum value and its derivatives. SECOA defines a one-way rolling function H, computing the digest of an input. To implement secure MAX aggregation, each sensor n_i computes a seed s_i from k_i. Suppose the reading of n_i is an integer v_i. n_i then applies H by v_i times on s_i (let the resulting digest be h_i) and

send the pair $< v_i, h_i >$ to the aggregator. On receiving all pairs from sensors, the aggregator computes the maximum value v_m. For any other value $v_i < v_m$, the aggregator further applies H by $v_m - v_i$ times on h_i and get the new digest, which is the result of applying H by v_m times on s_i. SECOA then defines folding function F that combines all digest into one, h. The aggregator sends the pair $< v_m, h >$ to the portal. Since the portal is aware of all k_i, it can compute the corresponding s_i and the digests by applying H v_m times on s_i. Using F to combine the digest together, the portal will get identical h if the aggregator did not cheat. Assume that node n_j has the maximum value v_m and the aggregator reports a value $v'_m < v_m$. Since the aggregator does not know k_j and H is applied in one-way, it is not able to generate the correct digest for n_j and h. Thus, by this one-way chain technique, the completeness of the aggregation is ensured in a communication-efficient way.

5.4 Efficient Algorithms for Specific Aggregations

As discussed in previous sections, COUGAR is a general framework for both acquisition and aggregation queries. TAG is optimized for aggregation queries, while TinyDB is acquisition-query oriented. CONCH further explores the spatial-temporal correlation among sensor readings to reduce the communication cost. However, for some specific aggregations, more efficient algorithms exist. This section surveys energy-efficient algorithms for continuous extreme and top-k value monitoring.

5.4.1 Extreme Value Monitoring. Extreme value monitoring in a WSN is extensively studied recently. TAG [8] supports MAX queries in a straightforward way. When a query comes, each leaf node in the routing tree sends its parent the current reading. An intermediate node sends its parent the maximum reading among all its children and itself. In the end, the maximum value propagates to the base station, which is the root of the routing tree. For a continuous query, such procedure is repeated in every cycle.

Rather than TAG, A. Silberstein et al. [11] propose a set of threshold-based algorithms for extreme value monitoring. The Hierarchical Adaptive Thresholds (HAT), which follows the tree topology, is the most energy-efficient. HAT maintains a threshold for each node, indicating the upper bound of the maximum value in its sub-tree. It is satisfied that a parent's threshold never falls below those of its children and the root's threshold is the current maximum value. For continuous queries, the root periodically issues signals requesting updates from the nodes.

On receiving the signal, a node sends its current reading to the parent. The benefit of the hierarchical thresholds is that when a reading with value v reaches a node having its threshold $\epsilon > v$, the transition stops. It is because the global maximum value, which equals to the threshold maintained at the root, is at least ϵ. Since v is less than ϵ, v cannot be the global maximum value.

5.4.2 Continuous Top-k Queries

. Top-k monitoring is a generalized version of the extreme value monitoring. Babcock et al. [30] address the problem of monitoring top-k values among distributed datasets. The idea is to align local top-k lists to global top-k list through some adjustment factors. However, the setting of the distributed datasets is quite different from that of the WSN applications. Wu et al. [29] propose FILA for top-k monitoring in WSNs. The basic idea is to install a local filter $[li, ui]$ at each node n_i, indicating that sensor readings of n_i do not affect the global top-k ranking if they fall in this filter and it is not necessary to report them. On the other hand, once a reading falls outside of the filter, it may affect the global top-k ranking and n_i reports to the base station n_0. n_0 maintains a synchronized copy for each filter. It reevaluates the filters once top-k values change and sends the new filters with corresponding nodes. By suppressing unnecessary updates, FILA outperforms straightforward approach implemented in TAG.

5.5 Join Processing

In-network JOIN operation (e.g. joining two records in the WSN that are within a specified time window) is not efficiently supported in TAG. The following shows a scenario of join in WSNs: In a volcano monitoring project, after noticing that the volcanic activity of the mountain has increased, scientists want to know whether the pressures detected have crossed a certain threshold and is continuously increasing within some period of time. An SQL-like query Q_6 is submitted to the network:

```
Q6:   SELECT   P_1.pressure, P_1.time, P_2.pressure, P_2.time
      FROM     Pressure AS P_1, Pressure AS P_2
      WHERE    P_1.pressure > threshold (ϵ)
      AND      P_2.pressure > P_1.pressure
      AND      P_2.time > P_1.time
      AND      P_2.time-P_1.time > h
```

where Pressure is the relation represents the sensor data on which self-join is performed. Specifically, two tuples of Pressure that fall in a time window are joined. Since a sensor does not know beforehand which nodes it may join with, a naive way is to let very node flood its tuples all over

the network, so as to discover possible joins partners. This causes huge communication cost which is a disaster for an energy critical setting. An alternative is let all nodes send their tuples to the base station, where JOIN operations are performed. Although reduced, the transmission amount of such approach is still considerable. Besides, the base station may become a bottleneck for massive data processing [36]. Yang et al. [18] propose Two-Phase Self Join (TPSJ), processing the above join in two phases. TPSJ is energy-effective and is applicable on queries having the following three properties:

1 The join involves two copies of the same relation.

2 The tuples joined are within a specific time window.

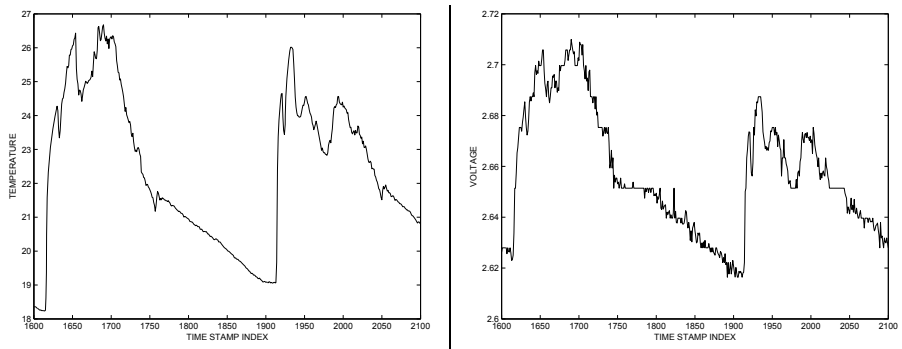
3 There is a selection predicate in the "WHERE" clause.

TPSJ decomposes the original query into two sub-queries, which are executed sequentially. As an example, the previous query Q_6 is decomposed into:

```
Q7:   SELECT   P.pressure, P.time INTO R_1
      FROM     Pressure AS P
      WHERE    P.pressure > threshold (ϵ)

Q8:   SELECT   P.pressure, P.time
      FROM     R_1, Pressure AS P
      WHERE    P.pressure > R_1.pressure
      AND      window(R_1.time, P.Time, h)
```

First, the base station issues Q_7 to the sensors. After Q_7 is executed, a table R_1, that contains all tuples satisfying the select predicates, is obtained at the base station. Then the base station issues Q_2. R_1 is also injected into the network along with Q_8. Since R_1 contains all join candidates, the correctness and completeness of the join results are ensured. The benefit TPSJ brings is to reduce the unnecessary transmissions of the useless tuples that do not join. The drawback is that the table R_1 has to be transmitted twice for one JOIN operation (first transmitted to the root and then injected into the network). Since the size of R1 is expected to be small, the overall reduction of transmission is substantial. Mihaylov et al. [44] summarize three classes of join strategies, i.e., the grouped join, through-the-base join and the pair-wise join. In the grouped join, the joined tuples are sent to a specific node using distributed/geographic hash table. In the through-the-base join, the tuples from one join party are routed to the join partner through the base station. In the pair-wise join, the algorithm first establishes a path between two join partners and then selects a node along this path to perform the join operation. Furthermore, the authors build cost models



(a) (b)

Figure 3.3. The correlation between different measurements

for the three join strategies, according to which the optimizer chooses the optimal query plan.

6. Other Queries

This section further investigates the model-driven data acquisition, probabilistic/approximation queries and event detection in WSNs.

6.1 Model-Driven Data Acquisition and Probabilistic Queries

In WSNs, correlations exist between different kinds of measurements. Figure 3.3(a) shows the temperature readings in a period of time, while Figure 3.3(b) shows the voltage level of a sensor in the same duration in the Intel Berkeley Data Set. It is obvious that there is strong correlation between the two measurements. The two curves are so similar that we can use one of them to predict the other. Another observation is that the energy cost of sampling temperature is much larger than that of retrieving the battery voltage. In order to reduce energy cost, instead of sampling temperature directly, we may first acquire the battery voltage and then predict the temperature reading. Motivated by this, Deshpande et al. [31] propose the model-driven data acquisition system called BBQ, a Tiny-Model Query System.

BBQ handles probabilistic queries. A probabilistic query typically includes two more parameters than the general queries:

1 . An error bound indicating how much bias from the real value is acceptable

2 . The confidence threshold of the returned value

A typically probabilistic query in BBQ is:

```
Q9:   SELECT   nodeID, temperature  0.1, confidence(0.95)
      FROM     Pressure AS P
      WHERE    P.pressure > threshold ϵ)
```

In the above query, the user asks for the temperature readings of nodes with ID from 1 to 8. The user specifies an error bound 0.1°C on temperature, meaning it is acceptable if the difference between the result r and the real value r* is no more than 0.1°. In addition, the user requires the probability that r* falls in the range $[r - 0.1°C, r + 0.1°C]$ is at least 95%, which is defined by the confidence parameter. BBQ builds a probability density function (pdf), for each attribute, according to the historical data. Suppose the sensor measures n attributes. The pdf is a function with n variables with the form $\mathrm{p}(X_1, X_2, \ , X_n)$. When a query on attribute att_i arrives, BBQ first marginalizes pdf with respect to att_i. It uses the marginalized pdf to compute the most possible value r of att_i, i.e., the expected att_i value. Then it calculate the probability pr that the real value of att_i falls in the range $[r - \epsilon, r + \epsilon]$, where ϵ is the given error bound. If pr satisfies the confidence parameter, r is returned. Otherwise, BBQ physically retrieves a new sample of att_i from sensors.

BBQ also handles probabilistic range queries. For instance, "check if the temperature at the 2nd node falls in the range $[20°C, 25°C]$". The procedure is similar. First, BBQ computes the marginalized pdf with respect to the temperature attribute. It then computes the probability that the expected temperature reading falls in the query range. If the probability is high (low) enough, BBQ is confident to returns true (false). Otherwise, BBQ is not confident enough to make the decision and it physically retrieves new samples from sensors to answer the query.

6.2 Event Detection

Event detection is an important application in WSNs. In forests, people use event detection to predict potential fire disasters. In factories, event detection is used to monitor abnormal machinery activities. Advised by the historical data, users of a WSN gain knowledge on the range of the sensor readings. For instance, in a factory, the temperature near machines is normally from $30°C$ to $60°C$ and the humidity in a barn is from 30% to 50%. If sensors report temperatures higher than $60°C$ or humidity below 30%, then somewhere in the factory is in danger of catching fire. *Condition-based* maintenance detects sensor readings fall out of the normal range, where conditions define the normal ranges of sensor readings. A condition consists of three attributes:

1 The attributes of the sensor, e.g., physical location and ID

2 The type of measurement, e.g., temperature and humidity

3 The range of the reading, e.g., temperature $[30°C, 60°C]$ and humidity $[30\%, 50\%]$

Tuples from related conditions form a table. For a complex monitoring application, there could be multiple condition tables. The goal is to collect sensor readings satisfying some condition predications [32]. As an example, consider the following scenario. In a factory, three machines are equipped with sensors for monitoring their temperatures. The condition tuple is in the form of $< MACHINE\_ID, range, time >$, indicating the range of normal temperature on different machines in different time slots. Table 3.2 shows a fraction of the condition table.

Table 3.2. A sample condition table

ID	Range	Time	ID	Range	Time
1	$[50°C, 70°C]$	$6am - 1pm$	2	$[20°C, 30°C]$	$6pm - 8am$
1	$[30°C, 60°C]$	$1pm - 7pm$	3	$[30°C, 60°C]$	$8am - 3pm$
1	$[10°C, 30°C]$	$7pm - 6am$	3	$[40°C, 70°C]$	$3pm - 10am$
2	$[60°C, 70°C]$	$8am - 6pm$	3	$[20°C, 40°C]$	$10pm - 8am$

Tuple $< 1, [50°C\ 70°C], 6am - 1pm >$ indicates that the normal temperature of the first machine varies in range $< 1, [50°C\ 70°C], 6am - 1pm >$] during 6am-1pm. If its temperature falls outside of the range during this time slot, it is abnormal and the system generates an event. Following shows a typical query which returns abnormal readings over all nodes:

```
Q10:   SELECT   a.temperature
       FROM     sensorAS a, condition_table AS t
       WHERE    a.temperature < t.min_temp
       AND      a.temperature > t.max_temp
       AND      a.nodeid = t.nodeid
```

The key in implementing this query is the JOIN operation that joins the condition table and the sensor readings. If the sensor memory is large enough for a condition table, it is straightforward to store the tables in selected nodes, retrieve them when tuple come and perform the join. However, it is always the case that the size of a condition table exceeds the sensor's memory capacity. Consequently, a condition table has to be split and stored multiple nodes. To tackle this challenge, Abadi et al. [32] propose an algorithm based on grouping sensors and partitioning condition tables. The idea is to horizontally partition a table and store the partial tables in a set of sensors. Due to table partitioning and distributed storage, a tuple t must be sent to all nodes containing the partial tables in order to complete the join. To make this procedure

communication-efficient, nodes are organize into groups. Sensors of the same group are within the broadcast range of each other, so that they can effectively exchange t and the partial join results. Protocols are proposed for group formation and join operation. Since connections in WSNs are unreliable, the protocols also detect sensor failures and develop recovering mechanism.

W. Xue et al. [33] defines events in a different way and develop corresponding detection techniques. Their work is motivated by coal mine applications. Events are defined by patterns of contour maps and are detected by matching the contour map of the current sensor readings with predefined patterns. A contour line of a function is a curve along which the function has a constant value [35]. In cartography, a contour line joins points of equal elevation above a given level, such as mean sea level. A contour map is a map consists of contour lines, for example a topographic map, which shows valleys and hills, and the steepness of slopes. Figure 3.4(a) illustrates how a contour map is created while Figure 3.4(b) shows a contour map of a hill.

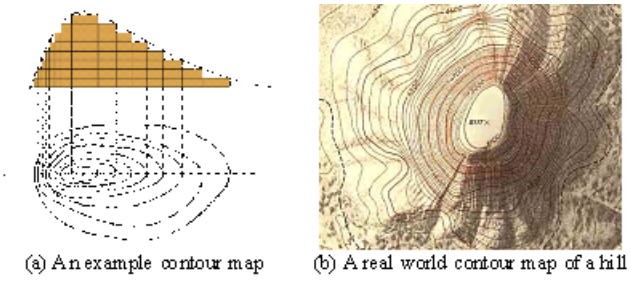


Figure 3.4. The Contour maps

In the coal mine application, the abnormal distribution of the oxygen or gas density and the humidity concern the researchers. The distribution of sensor readings (e.g., gas density) over the monitoring area is represented by a contour map. The authors observe that when some events happen, the shapes of the contour maps exhibit certain patterns. [33] follows the tree topology. During transmission, a sensor generates its partial contour map by combining all partial maps from its children. Since contour maps are typically of large size, the authors propose compressing techniques to optimize the map transmission. Another observation is that the contour maps are mostly stable over time. For continuous event monitoring, instead of transmitting contour maps at each time stamp, only the updates upon the previous ones are transmit-

ted. Maps are updated incrementally at the base station, which matches the maps against the predefined patterns and reports the events.

6.3 Approximation Queries

Approximation is a common approach to achieve data reduction if no exact answers are required. In applications such as military surveillance, in order to achieve high responsiveness, it is desirable that the system quickly returns a coarse result for the overall battlefield, and then refines the result on specific areas iteration by iteration, similar to zoom in operation on image viewing. [42] proposes data shuffling algorithm to tackle this problem. The purpose of data shuffling algorithm is to determine the reporting order of sensor readings so that the algorithm is able to compute the approximated result from the first few reports and then refine the result on receiving the additional reports. [43] proposes another approach, called the multi-resolution compression and query (MRCQ). In MRCQ, sensors are conceptually organized in a layered structure where a node may work in multiple layers. The raw readings are stored at the bottom layer in the form of matrices. A bottom layer node applies the discrete cosine transform on its matrix and gets i) the compressed representation of its matrix and ii) the stored data. The major characteristics of the matrix are stored in i), which is transferred to the upper layer node as the approximation, i.e., the low resolution result. In order to recover the original data, both i) and ii) are necessary. In case refinements are required later, a bottom node caches the stored data locally. The same procedures are repeated in the intermediate nodes, until the compressed data reach the base station. In order to refine the query result, the base station requests the stored data cascading down. MRCQ achieves different resolutions of the result by stopping the cascading at different levels of the layered structure.

7. Conclusion

In this chapter, we survey the recent works in WSNs under a database point of view. Starting by investigating the characteristics of sensors, it addresses the common issues in designing algorithms in the WSNs. It follows to introduce the network topologies of WSNs, which are the foundation of data transmission. Then we investigate the general data acquisition and aggregation frameworks and shows the common query languages and query types in WSNs. More specific applications, query types, and various approaches are also discussed.

References

[1] Yick, J., Mukherjee, B., Ghosal, D. *Wireless sensor network survey.* In Computer Networks 52(12):2292-2330, 2008.

[2] Juang, P., Oki, H., Wang, Y., Martonosi, L., Peh, S., Rubenstein, D. *Energy Efficient Computing for Wildlife Tracking: Design Tradeoff and Early Experiences with Zebranet.* In Int. Conf. on Architectural support for Prog. Languages and Op. Sys., 2002.

[3] Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A., Estrin, D. *Habitat monitoring with sensor networks.* In Communications of the ACM, 2004.

[4] Madden, S., Franklin, M., Hellerstein, J. *The Design of an Acquisitional Query Processor for Sensor Networks.* In ACM SIGMOD, 2003.

[5] Abadi, D., Madden, S., Lindner, W. *REED: Robust Efficient Filtering and Event Detection in Sensor Networks.* In VLDB, 2005.

[6] Nakamura, E., Loureiro, *A. Information Fusion in Wireless Sensor Networks.* In ACM SIGMOD, 2008.

[7] Madden, S., Franklin, M., Hellerstein, J., Hong W. *TinyDB: An Acquisitional Query Processing System for Sensor Networks.* In ACM Transactions on Database Systems, 2004.

[8] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E. *Wireless sensor networks: a survey.* In Computer Networks, 2002.

[9] Manjhi, A., Nath, S., Gibbons, B. *Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams.* In ACM SIGMOD, 2005.

[10] Silberstein, A., Braynard, R., Yang, J. *Constraint Chaining: On Energy-Efficient Continuous Monitoring in Sensor Networks.* In ACM SIGMOD, 2006.

[11] Silberstein, A., Munagala, K., Yang, J. *Energy-Efficient Monitoring of Extreme Values in Sensor Networks.* In ACM SIGMOD, 2006.

[12] Considine, J., Li, F., Kollios, G., Byers, J. *Approximate Aggregation Techniques for Sensor Databases.* In ICDE, 2004.

[13] Madden, S., Franklin, M., Hellerstein, J., Hong, W. *TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks.* In OSDI, 2002.

[14] Nath, S., Gibbons, P., Seshan, S., Anderson, Z. *Synopsis Diffusion for Robust Aggregation in Sensor Networks.* In SenSys, 2004.

[15] Intanagonwiwat, C., Govindan, R., Estrin, D. *Directed diffusion: A scalable and robust communication paradigm for sensor networks.* In ACM MobiCOM, 2000.

[16] Intanagonwiwat, C., Estrin, D., Govindan, R., Heidemann, J. *Impact of network density on data aggregation in wireless sensor networks.* In ICDCS, 2002.

[17] Woo, A., Culler, D. *A transmission control scheme for media access in sensor networks.* In ACM Mobicom, 2001.

[18] Omotayo, A., Hammad, M., Barker, K. *A Cost Model for Storing and Retrieving Data in Wireless Sensor Networks.* In ICDE, 2007.

[19] Dantu, R., Abbas, K., II, M., Mikler, A. *Data centric modeling of environmental sensor networks.* In IEEE Global Telecommunications Conference Workshops, 2004.

[20] Ganesan, D., Estrin, D., Heidemann, J. *Dimensions: Why do we need a new data handling architecture for sensor networks.* In Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), 33(1):143-148, 2002.

[21] Ghose, A., Grossklags, J., Chuang, J. *Resilient datacentric storage in wireless ad-hoc sensor networks.* In Proceedings of the 4th International Conference on Mobile Data Management, 2003.

[22] Gummadi, R., Li, X., Govindan, R., Shahabi, C., Hong, W. *Energy-efficient data organization and query processing in sensor networks.* In 21st International Conference on Data Engineering, 2005.

[23] Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Li, Y., and Yu, F. *Data-centric storage in sensornets with GHT, a geographic hash table.* In Mobile Networks and Applications, 8(4):427-442, 2003.

[24] Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., Estrin, D. *Data-centric storage in sensornets.* ACM SIGCOMM Computer Communication Review, 33(1):137-142, January 2003.

[25] Li, X., Kim, Y., Govindan, R., Hong, W. *Multidimensional range queries in sensor networks.* In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, 2003.

[26] Seshadri P., Livny M., Ramakrishnan R.: *SEQ: A Model for Sequence Databases.* In ICDE, 1995.

[27] Bonnet P., Gehrke J., Seshadri P. *Towards Sensor Database Systems.* In Mobile Data Management, 2001.

[28] *The COUGAR project.* `http://www.cs.cornell.edu/bigreddata/cougar/index.php`.

[29] Wu, M., Xu, J., Tang, X., Lee, W. *Top-k Monitoring in Wireless Sensor Networks.* In IEEE TKDE, 2007.

[30] Babcock, B., Olston, C. *Distributed Top-k Monitoring.* In ACM SIGMOD, 2003.

[31] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., Hong, W. *Model-Driven Data Acquisition in Sensor Networks.* In VLDB, 2004.

[32] Abadi, D., Madden, S., Lindner, W. *REED: Robust, Efficient Filtering and Event Detection in Sensor Networks.* In VLDB, 2005.

[33] Xue, W., Luo, Q., Chen, L., Liu, Y. *Contour map matching for event detection in sensor networks.* In SIGMOD, 2006.

[34] Deligiannakis, A., Kotidis, Y., Roussopoulos, N. *Compressing Historical Information in Sensor Networks.* In SIGMOD, 2004.

[35] *Contour line.* `http://en.wikipedia.org/wiki/Contour_line`.

[36] Yang, X., Lim, H., Özsu, M., Tan, K. *In-network execution of monitoring queries in sensor networks.* In SIGMOD, 2007.

[37] *The Crossbow Technology.* http://www.xbow.com/.

[38] Madden, S., Szewczyk, R., Franklin, M., Culler, D. *Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks.* In IEEE WMCSA, 2002.

[39] Kim, S., Fonseca, R. *Reliable Transfer on Wireless Sensor Networks.* In IEEE SECON, 2004.

[40] Jindal, A., Psounis, K. *Modeling spatially correlated data in sensor networks.* In ACM Transactions on Sensor Networks, 2(4):466-499, 2006.

[41] Nath, S., Yu, H., Chan, H. *Secure outsourced aggregation via one-way chains.* In SIGMOD, 2009.

[42] Liu, Y., Li, J., Fang, X. *Enabling Approximate Querying in Sensor Networks.* In VLDB, 2009.

[43] Wang, Y., Hsieh, Y., Tseng, Y. *Multi-Resolution Spatial and Temporal Coding in a Wireless Sensor Network for Long-Term Monitoring Applications.* In IEEE Trans. Computers, 58(6):827-838, 2009.

[44] Mihaylov, S., Jacob, M., Lves, Z., Guha, S. *Dynamic Join Optimization in Multi-Hop Wireless Sensor Networks.* In PVLDB, 2010.

Chapter 4

EVENT PROCESSING IN SENSOR STREAMS

Fusheng Wang
Emory University
Atlanta, GA, USA
fusheng.wang@emory.edu

Chunjie Zhou
Ludong University
Yantai, China
lucyzcj@163.com

Yanming Nie
Northwest Sci-Tech University of Agriculture and Forestry
Yangling, China
yanmingnie@nwsuaf.edu.cn

Abstract Sensors including RFID tags have been widely deployed for measuring environmental parameters such as temperature, humidity, oxygen concentration, monitoring the location and velocity of moving objects, tracking tagged objects, and many others. To support effective, efficient, and near real-time phenomena probing and objects monitoring, streaming sensor data have to be gracefully managed in an event processing manner. Different from the traditional events, sensor events come with temporal or spatio-temporal constraints and can be non-spontaneous. Meanwhile, like general event streams, sensor event streams can be generated with very high volumes and rates. Primitive sensor events need to be filtered, aggregated and correlated to generate more semantically rich complex events to facilitate the requirements of up-streaming applications. Motivated by such challenges, many new methods have been proposed in the past to support event processing in sensor event streams. In this chapter, we survey state-of-the-art research on event processing in sensor networks, and provide a broad overview of major topics in

complex RFID event processing, including event specification languages, event detection models, event processing methods and their optimizations. Additionally, we have presented an open discussion on advanced issues such as processing uncertain and out-of-order sensor events.

Keywords: Sensor streams, RFID streams, event processing

1. Events and Event Processing

We first present an overview of events and event processing in the context of sensor streams, including semantics of events, event processing, and use cases of sensor event processing.

1.1 Semantics of Events

An "***event***" is a happening of interest [77]. In database applications, the interest in events comes mostly from the state changes that are produced by data manipulation operations [54]. Example events in real world include a financial trade, a web click, a sensor reading and a social or natural significant happening, and many others. In a monitored environment deployed with sensors, flows of observation data can be seen as streams of observable events. When an event takes place, we refer to its ***occurrence***; and when an event is recognized by the system, we refer to its ***detection***.

Events are often interrelated and form complex relationships, such as temporal, spatial, causal or abstract (or composite) relationships. A temporal, spatial or causal relationship of events can determine the partial order between events, and abstract or composite relationship can be represented as an event that represents or summarizes a collection of events.

Events can be categorized as atomic events and composite events. ***Atomic events*** or ***primitive events*** are the simplest events in a system, which are defined to occur at a certain time point or not occur at all. A ***composite event*** or ***complex event*** is a high-level derived event, and it is defined by applying an event operator to constituent events that are primitive events or other composite events [54].

Events can have their attributes such as type, ID, and time; an event attribute can have a simple or complex data type. Similar events can be grouped into an event type, and an event type is denoted by an event expression. A primitive event (type) name itself is an event expression. If E_1, E_2, ..., E_n are event expressions, an application of any event operator over the event expressions is an event expression. An atomic

event is defined to occur at a point of time, usually called as a ***point event***; while a composite event can always span a period of time, i.e., an ***interval event*** [72]. Any dimension of attributes of an event can be either certain or uncertain. An event with one or multiple uncertain attributes is an ***uncertain event*** or a ***probabilistic event*** [53]; otherwise, it is a ***certain event***. If an event cannot detect its occurrence by itself unless it either gets expired or is explicitly queried, we name it as a ***non-spontaneous event*** [73]. RFID applications and sensor applications can generate non-spontaneous events such as negated events and temporal constrained events. Such non-spontaneous events pose new challenges for event processing.

It is common that real-world events are associated with time and spatial or location dimensions, which mirror the most common inquiries about such events, i.e., *when* and *where*. However, events can contain more information than these two well-known dimensions entail. Originally, other semantic properties such as genealogy, identification and others can describe partitional information of an event. All these properties of an event can be viewed as its context, namely ***event context***, and event context can be temporal, spatial, semantic, or even social. Contexts of events can significantly affect the semantics for event processing, and it is critical to identify the context and the type of context to process events effectively and semantically.

1.2 Event Processing

Vast amounts transaction data and monitoring data can be constantly generated as event streams, which have to be fully processed to support automated business decisions or time-critical actions. Basically, events cannot be entirely foreseen [62], and we cannot predict whether a critical event will happen, or when it will happen. In reality, what we can do is to ensure that the interested events can be detected in a real-time or quasi-real-time manner; this is the main purpose of ***event processing***. Thus timeliness is among the top priorities in event processing applications.

Generally speaking, event processing can be broadly defined to be any computing that performs operations on events, including reading, creating, transforming and deleting events [28]. The main idea of event processing is to process events to gather meaningful or valuable information and then deriving actions from them. The main functional capabilities required by event processing applications include data filtering, aggregation, transformation, pattern detection, pattern discovery and pattern prediction. Non-functional requirements include performance, response time and throughput.

A strong connection exists between the ***event-condition-action*** (ECA) paradigm [64] and event processing (EP). EP applies three basic ECA concepts, i.e., events, conditions and actions. However, EP considers more complex events, conditions, and actions, and traditional ECA rules are insufficient for more complex conditions and actions. The differences between ***stream processing*** (SP) [65] and event processing (EP) are some blurry. Both SP and EP have the ability to to efficiently process long-running continuous queries over sequences of events. SP tends to place a higher emphasis on managing large volumes of data with relatively fewer queries; whereas EP tends to consider the effect of sharing across many queries or many patterns and focus on response generation [63].

Complex event processing (CEP) [66] refers to effective detection and evaluation of the specified meaningful event patterns such as opportunities, exceptions, or threats over event streams. It is also often referred to as complex pattern matching. The goal of CEP is to identify meaningful events such as opportunities or threats and generate timely responses. Based on Mythbusters [67], EP is analogous to signal processing and CEP is more aligned with higher level situational inferencing.

1.3 Applications of Sensor Event Processing

In the era of smart planet, a variety of sensors including RFID have been widely deployed within wired or wireless networks to produce measurements and observations. The sensor data can be viewed as events (i.e., ***sensor events***) and can be utilized for the purposes of probing and monitoring. Sensor network applications can be categorized into areas such as military intelligence, environment monitoring, municipal administration, industry production, health-care assistance, smart home and so on. Next we summarize some common event-driven sensor-enabled example applications, which generates huge volume of observing events to be handled with EP mechanisms.

Military Intelligence Applications: Military intelligence is inherently based on an information-rich environment. Nowadays, the number of sensors, satellites, and soldiers is pervasive, and the need to present a timely, correct, aggregated and integrated view based on the multiple sensing information sources is critical for effective and precise decision making.

Environment Monitoring Applications: Multiple sensors can be deployed at different sites on the mountain-side for the prediction of geological disasters such as avalanche at the monitored mountain. There are several influential parameters including static ones such as the steepness

of the mountain and dynamic ones such as temperature, air pressure and snow depth. The measurements and their history values play a critical role for correct and timely event processing.

Municipal Administration Applications: A traffic monitoring system can gather GPS data transmitted by cars, including the ID, position, and speed of cars. Besides, vast amount of sensors can also be deployed at main routes and key junctions to collect real-time traffic conditions. Each car transmits a packet of data periodically. The monitoring system can introduce event processing to correlate the data in order to detect traffic problems, recommend ideal driving routes, or send route in real-time.

Industry Production Applications: Subtle electronic products such as chips should be manufactured in a finely controlled production environment. Many conditions of the plants such as the vibration of equipment parts need to be carefully monitored. In such case, corresponding sensors which measure mechanical vibrations can be used to avoid inefficient manual measuring and patrolling. Recently, RFID has been widely adapted for tracking the exact locations of items or tallying the number of items in the same category in a modern warehouse.

Health-care Assistance Applications: Sensors are often deployed to track and monitor doctors and patients inside a hospital. We can imagine such a scenario: Each patient can be attached a small and light weight sensor node for detecting the heart rate or the blood pressure. Doctors may also carry sensor nodes so their location inside the hospital can be quickly identified. Similarly, sensors can be mounted at some critical areas to facilitate more effective monitoring. For the sake of reasonable diagnosis or prediction, sensor events from multiple monitors should be correlated, and knowledge of the patient's condition and disease history should also be considered during this type of sensor event processing.

Smart Home Applications: For home automation, smart sensor nodes and actuators can be embedded in appliances such as vacuum cleaners, microwaves, ovens, refrigerators, and VCRs. Sensors for measuring temperature, humidity, percentages of air components and others can also be deployed in the home to monitor indoor conditions or actuate adjustments through smart appliances such as air conditioners. These sensor nodes can interact with each other, and can be conveniently monitored and managed by end users through via the Internet or satellites.

To support effective, efficient, and near real-time phenomena probing and objects monitoring, streaming sensor data have to be gracefully managed in an event processing manner. Different from the traditional events in active databases, sensor events come with temporal or even

spatio-temporal constraints and can be non-spontaneous. Meanwhile, like general event streams, sensor event streams can be generated with very high volumes and rates. Primitive sensor events need to be filtered, aggregated and correlated to generate more semantically rich complex events to facilitate the requirements of up-streaming applications. Motivated by such challenges, many new methods have been proposed in the past to support event processing in sensor event streams. In this chapter we will mainly focus on complex event processing over sensor streams including RFID data streams, as the trends show that primitive event processing is gradually moved to the edges of event sources. Next we will present an overview of sensor event processing techniques, including event specification languages, event detection models, event processing methods and their optimizations. During the discussion, we will pay special attention to the distinct challenges of event processing over sensor streams and RFID streams.

2. Event Processing in Sensor Streams

Event detection approaches in sensor networks can be categorized into statistical methods [1], topographical techniques [2–4], and edge detection algorithms [5–7].

Statistical methods. A statistical method is presented in [1] for detecting generic homogeneous regions without the benefit of an a priori predicate to identify events. Instead, it uses a kernel density estimator to approximate the probability density function of the observations. It is suggested that the detection routine be rerun periodically to accommodate the scenario of any new regions or holes that evolve in the midst of tracking. Even so, there is not an elegant way to handle new detections and persistent tracking at the same moment.

Topographical methods. An example of the topological and contour mapping technique is Iso-Map [2], which builds contour maps based solely on the reports collected from intelligently selected "isoline nodes" in the network. This approach is limited to a plane. Another technique [3] collects time series of data maps from the network and detects complex events through matching the gathered data to spatio-temporal data patterns. Essentially the work provides a basic infrastructure and then outsources the problem solution to the user, instead of directly solving the event tracking problem. SASA [4] uses a hole detection algorithm to monitor the inner surface of tunnels, where sensor nodes may be displaced due to collapses of the tunnels.

Edge detection methods. In edge detection based event detection and tracking, the challenge is to devise a method for nodes to be identified as "edge nodes" that are near the boundary of a region, and from that, calculate an approximate boundary for the region in question. Three methods guided by statistics, image processing techniques, and classifier technology are developed and compared in [5]. A novel method for edge detection of region events makes use of the duel-space principle [6, 7]. The algorithm is fundamentally centralized, but it can be distributed among backbone nodes in a two-tier architecture. This approach, like [5], does not accomplish event labeling.

Existing research on point event detection includes various protocols such as Distributed Predictive Tracking [8], Dynamic Convoy Tree-based Collaboration (DCTC) [9] and theoretical contributions [10]. One of the most notable contributions is DCTC [9]. It uses a "Dynamic Convoy Tree" protocol to accomplish both event tracking and communication structure maintenance. DCTC essentially forms and maintains a spanning tree over the nodes which senses the event. This is perhaps the most obvious and straightforward method of detecting events within the network. Moreover, many of the existing high level event detection services either cite DCTC directly or at least assume a spanning tree structure like it as part of the middleware needed for their query support.

2.1 Event Models for Sensor Streams

Next we consider two application scenarios of event models for sensor streams. The first is an offline variant in which event detection happens at the database that stores the measurements collected by the network. This detection method is used to automatically identify "interesting" regions within the swaths of data acquired by the sensor network. In the other online application, motes in the network use events and models to alter their behavior.

Offline event detection. The offline event detection provides a model suitable for querying events from noisy and imprecise data. Both database systems [12, 13] and sensor networks [14–16] have explored model-based queries as a method for dealing with irregular or unreliable data. Models in these systems include Gaussian processes [14], interpolation [17, 18], regression [14–19] and dynamic-probabilistic models [13–15]. PCA (Principal Component Analysis) based model is specifically suited to event detection [11]. MauveDB [13] provides a user-view interface to model-based queries, which greatly extends the utility and usability of models.

Online event detection. In the online case, sensor networks reduce the bandwidth requirements of data collection by suppressing results that conform to the model or compressing the data stream through a model representation. This has coincident benefits on resource and energy usage within the network. If sensors measure spatially correlated values, values collected from a subset of nodes can be used to materialize the uncollected values from other nodes [20, 21]. Similarly, temporally-correlated values may be collected infrequently and missing values can be interpolated [15, 22]. By placing models in the mote itself, the mote may transmit model parameters in lieu of the data, compressing or suppressing entirely the data stream [23–25].

There is also work on defining a common conceptual model of event processing based on event driven architectures [27] and event processing networks [28]. In PCA model [11], the notion of event history or event flow is different from those used in [19, 20] such that the event history flow takes embedded uncertainty. In fact it contains observations (event clusters), which consist of multiple possible events. In those models an event history itself is considered deterministic and the uncertainty on event history is expressed as there can be multiple possible event histories. Due to this difference, the rule semantics is totally different from the conditional representation in [19]. Ganeriwal et al. [26] discuss the reputation-based framework for high integrity sensor networks. The model evaluates the trustworthiness of the nodes and various misbehavior types of nodes in the network. The model uses the Bayesian formulation and updates the trust with direct and indirect trust calculations.

2.2 Sensor Event Detection

Much work has been done in sensor networks on composite event detection. Directed Diffusion [29] is among the earliest event-based approaches. In this approach, a node would request data by sending interests, which is conceptually similar to subscriptions in a publish/subscribe system. Data found to match those interests are then sent towards that node. A different framework based on event classification is the Online State Tracking [30] approach. This technique consists of two phases: the first phase is the learning process where new sensor readings are classified to states, and the second phase is the online status monitoring phase where nodes are collaborating to update the overall status of the network. The work is quite unique in the sense that it moves away from individual node readings and views the whole network as a state machine.

Another event-based technique based on threshold is Approximate Caching [31] whereby nodes only report readings if they satisfy a condition. A more recent paper [32] suggests a mixture of hardware and software as a solution for detecting rare and random events. The event types they consider are tracking and detecting events using the eXtreme Scale Platform (XSM) motes equipped with infrared, magnetic and acoustic sensors. Central to their architecture is the concept of passive vigilance, which is inspired from sleep states of humans where the slightest noise can wake us up when we are asleep. This is implemented with Duty Cycling and recoverable retask. A similar approach [33] proposes a sleep-scheduling algorithm that minimizes the surveillance delay (event detection delay) while it maximizes energy conservation. Sleep scheduling is coordinated locally in a fair manner, so all nodes get their fair share of sleep. A minimal subset that ensures coverage of the sensing field is always awake in order to be able to capture rare events.

The earliest work that addresses the need for complex event detection is the one by Girod et al [34]. It suggests a system that would treat a sequence of samples (a signal segment) as a basic data type and would offer a language (WaveScript) to express signal processing programs as declarative queries over streams of data. The language would be able to execute both on PCs and distributed sensors. The data stream management system (called WaveScope) combines event-stream and data management operations.

REED [35] is an approach that falls under both the Event-Based and the Query-Based subcategories. REED is an improvement on TinyDB [36]. Basically it extends TinyDB with the ability to support joins between sensor data and building static tables outside the network. The tables outside the network describe events in terms of complex predicates. These external tables are joined with the sensor readings table, and returned tuples that satisfy the predicates indicate readings of interest, for example, where an event has occurred.

Abstract Regions [37, 38] is a somewhat different method that supports geographic grouping of sensor nodes. Abstract Regions is essentially a family of spatial operators for TinyOS that allows nodes to form groups with the objective of data sharing and reduction within the groups by applying aggregate operators such as min, max, sum, and others. The work by [39] extends the types of aggregates supported by introducing (approximate) quantiles such as the median, the consensus, the histogram and range queries. Support for spatial aggregation is also suggested by [40] where sensor nodes would be grouped and aggregates would be computed using Voronoi diagrams. Another approach [41] models the sensor network as a distributed deductive declarative

database system. The method allows for composite event detection, and the declarative language used (SNLog) is a variant of Datalog.

3. Event Processing over RFID Streams

One essential goal for RFID applications is to map objects and their behaviors in the physical world into the virtual counterparts and their virtual behaviors in the applications by semantically interpreting and transforming RFID data. Application logic can often be devised and engineered as complex RFID events, and once such complex events are detected, the semantics can be automatically interpreted. Based on the purposes of RFID data processing, RFID applications can be generally classified as two categories: i) history-oriented object tracking supported through temporal database or data warehousing based solutions [75, 76], and ii) real-time oriented monitoring and stream processing through complex RFID event processing techniques. Complex RFID event processing plays a critical role on interpreting the semantics of RFID data and supporting real-time monitoring applications.

Basic theory of complex event processing has been intensively studied in the area of active database. There exist several processing models, including automata-based, Petri net- based, matching tree- based and directed graph- based. As these processing models did not fully consider the characteristics and complex semantics of RFID events, they can not be applied to RFID complex event processing immediately.

Different from the events in traditional active databases [54, 55] and message-based processing systems [56], RFID events have their own unique characteristics. First, RFID events are temporally constrained: both the temporal distance between two events and the interval of a single event are critical for the event detection. In addition, RFID applications can also generate non-spontaneous events - events that cannot detect their occurrences by themselves unless they either get expired or are explicitly queried. These include negated events (an event which does not occur) and temporal constrained events, for example, an event that occurs within a certain period. Such non-spontaneous events pose new challenges for event processing. Moreover, the actions from RFID events are quite different: they are normally database updates and messages, and neither trigger new primitive events for the system, nor lead to a cascade of rule firings as in active databases.

Some large-scale IT application providers [57–60] and academic institutions [61] had provided many platforms to collect data from RFID readers and pump the collected data to upper down-streaming systems. However, these platforms currently only support simple event process-

ing, for example, filtering or simple composition of primitive events, yet can not be used for answering complex queries.

There is much work on complex RFID event processing, and representative ones include SASE [42] and RCEDA [73]. Next we will describe in detail on the two frameworks in terms of event specification languages, event detection models, event detection methods and their optimizations.

3.1 RFID Events

An RFID event is an occurrence of interest in time, and it can be either a primitive event or a complex event.

A *Primitive RFID event* (also as atomic RFID event) is an event generated during the interaction between a reader and a tagged object. A primitive event is simple-semantic and represented as a triplet with the form of `observation(r, o, t)`, where r represents the reader EPC, o represents the object EPC and t represents the observation timestamp. The Electronic Product Code is an industry standard that defines unique code to identify an object around the world. The unique identification of each tagged RFID object through EPC code provides more semantics for RFID events.

For example, `observation(r1, o1, t1)` represents an event generated for an object with EPC *o1* from a reader with EPC *r1* at time *t1*. Primitive events are instantaneous, i.e., given any primitive event e, its staring time equals to its ending time. Primitive events are also atomic: a primitive event either happens completely or does not happen at all.

A *complex RFID event* or composite RFID event is usually defined by applying event constructors to its constituent events, which are either primitive events or other complex events. There are two types of RFID event constructors: non-temporal and temporal, and the latter contains order, temporal constraints, or both. While complex events defined with non-temporal event constructors can be detected without considering the orders among constituent events, complex events defined with temporal event constructors cannot be detected without checking the orders, or other temporal constraints among their constituent events, or both.

For example, shoplifting can be represented as a complex RFID event: an item was picked at a shelf and then taken out of the store without being checked out. This complex RFID event consists of three primitive RFID events: two occurrences of the tagged item being detected at a shelf and the exit respectively, and the non-occurrence of the item being read by any check-out reader in-between.

3.2 RFID Complex Event Specifications

Major RFID event processing frameworks use expressive languages to specify complex RFID events. As an RFID complex event specification language, SASE [42, 68] language is SQL-like, and supports sequencing, negation operation (!), parameterized predicates and sliding windows. The SASE language can be used to filter, correlate and transform primitive RFID events into complex events to answer semantic queries. The overall structure of SASE language is:

EVENT <event pattern>
[WHERE <qualification>]
[WITHIN <window>]

The *EVENT* clause describes a sequence pattern, and its components are occurrence or non-occurrence of component events in a temporal order. The *WHERE* clause specifies constraints on those events. The *WITHIN* clause specifies the sliding window for the whole sequence of events. For example, the complex event corresponding to shoplifting in a retail store can be specified as Q_1:

Q_1: *EVENT SEQ(SHELF x, !(COUNTER y), EXIT z)*
WHERE x.Oid=y.Oid=z.Oid
WITHIN a hour

In Q_1, *SEQ* denotes sequence pattern. *SHELF*, *COUNTER* and *EXIT* are different event types. The sign '*!*' denotes non-occurrence of an event (also called as a negation event).

To enhance its expressibility and adaptability, the SASE language has been extended to support Kleene closure [69]. NEEL [71] is a complex event specification language for the definition of embedded sequences (or nested sequences) of RFID events, and is essentially an extension of the SASE language.

Based on Snoop [72], an expressive event specification language for active databases, Wang and et al [73, 74] formalize the semantics and specification language of RFID events, and propose powerful rules for RFID data filtering, transformation, aggregation, and real-time monitoring. For clarity and convenience, here we refer to this specification language as RCEDA (i.e., RFID Complex Event Detection Algorithm) language.

The RCEDA language defines three basic non-temporal constructors including `OR` ($\vee$), `AND` ($\wedge$) and `NOT` ($\neg$), and five temporal constructors including `SEQ(;)`, `TSEQ(:)`, `SEQ`$^+$`(;`$^+$`)`, `TSEQ+(:`$^+$`)` and `WITHIN`. Most temporal constructors come with two types of temporal constraints: the

distance constraint and the interval constraint. Such fundamental event constructors can be combined to form complex RFID events.

For example, a company uses RFID tags to identify asset items and employees in the building, and only authorized users (superusers) can move the asset items out of the building. When an unauthorized employee or a criminal takes a laptop (with an embedded RFID tag) out of the building, the system will send an alert to the security personnel for response. Such complex RFID event pattern (Q_2) can be expressed in the RCEDA language as follows:

WITHIN(E1 ∧ ¬ E2, 5sec)

Here events `E1` and `E2` are two primitive events:
`E1 = observation('r2', o1,t1), type(o1)= 'laptop'`
`E2 = observation('r2', o2, t2), type(o2) = 'superuser'`.

Based on the event specification described above, RFID rules can be defined to support data filtering, data transformation, data aggregation and real-time monitoring. The RFID rule for event Q_2 is shown below:

DEFINE E4 = observation('r4', o4, t4), type(o4) = 'laptop'
DEFINE E5 = observation('r5', o5, t5), type(o5) 'superuser'
CREATE RULE r5, asset_monitoring_rule
ON WITHIN(E4 ∧ ¬ E5, 5sec)
IF true
DO send_alarm

Here *r5* and *asset_monitoring_rule* are unique rule id and rule name respectively. `WITHIN`*(E4 ∧ ¬ E5, 5sec)* is the event part of the rule. *send_alarm* is an action to be performed while the specified event occurs. According to the defined rule, an alert alarm will be issued when an unauthorized employee takes a laptop out of the building.

3.3 RFID Complex Event Detection Models

While RFID event specification languages provide an expressive way to specify complex RFID events, the detection of such events is much more challenging. The detection models in active databases have limitations on supporting RFID events. Automata-based model [77] and the PetriNet-based model [78, 79] require that all the timestamps of the constituted events are in total order. Tree-based model [54] and graph-based model [80] does not support time constraints. All these traditional models can not be directly used for RFID complex event detection.

An NFA-based complex RFID event detection model supplemented with Partitioned Active Instance Stacks (PAIS) [42] is proposed to support complex RFID event detection, especially for event backtracking

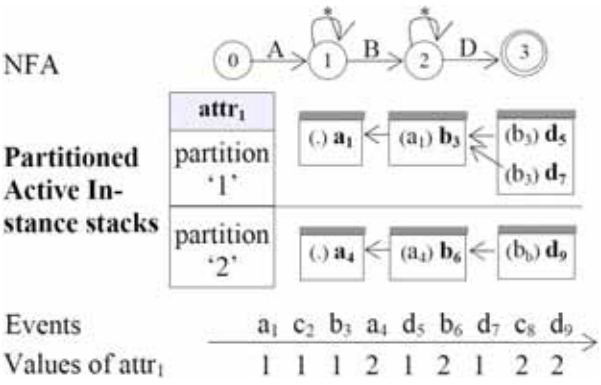


Figure 4.1. An NFA structure and PAIS for sequence Q3.

and value constraint evaluation during the process of complex event detection. For example, Figure 4.1 shows the NFA structure and the PAIS for Query Q_3 `(SEQ(A, B, D))`.

(`SEQ(A, B, D)`) is illustrated in Figure4.1.

A SASE extended model which combines a finite automaton with versioned match buffers is proposed in [70] to support event backtracking and value constraint evaluation during complex RFID event detection and general pattern matching.

Traditional tree based event detection modes take an bottom-up approach (e.g., Snoop [72]), which is inapplicable to detecting RFID events. Many temporal constrained RFID events such as those generated from `SEQ`$^+$ and `NOT` constructors are non-spontaneous and can never be triggered by the bottom-up approach. As summarized in [73, 74], there are three event detection modes such as Pull ($\uparrow$), Push($\downarrow$) and Mixed ($\updownarrow$) generalized in the RCEDA framework.

RCEDA model [73, 74] extends tree-based detection model for temporal constraints handling. Fundamental event constructors form basic tree operators (Figure4.2), and complex events can be represented by combining these tree operators to form more complex tree based representations. For example, Figure 4.3b illustrates the graphical representation of a complex event `E = WITHIN(TSEQ`$^+$`(E1`$\vee$`E2, 0.1sec, 1sec) ; E3, 10min)` after interval propagating from Figure 4.2. We use `vE.within` to represent the interval constraint on event *E*. To support both pull and push modes, RCEDA provides two way detections through the tree model: bottom-up event propagation through the tree to trigger parent events, and top-down event querying to support the detection of non-spontaneous events. The detection of non-spontaneous events is supported through the introduction of "***pseudo-events***". A pseudo event

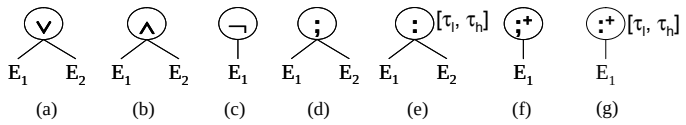


Figure 4.2. Graphical representations of the seven complex event constructors.

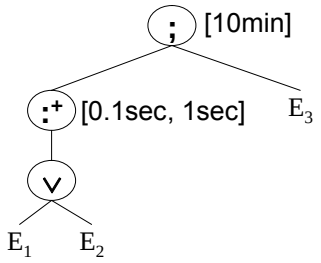


Figure 4.3. Graphical representations of an interval-constrained complex event E = WITHIN(TSEQ$^+$(E1∨E2, 0.1sec, 1sec); E3, 10min)

is a special artificial event used for querying the occurrences of non-spontaneous events during a specific period, and is scheduled to happen at an event node's expiration time.

3.4 RFID Complex Event Detection Methods and Optimizations

Based on the defined specification language for complex RFID events, the SASE framework [42] takes a query plan based method for complex RFID event detection. SASE defines several operations including temporal relationship, numerical constraints, negation and sliding window, and six operators such as Sequence Scan and Construction (SSC), Selection (σ), Window (WD), Negation (NG) and Transformation (TF). These operations and operators are used to form complex RFID event query plan in a bottom-up manner. An illustration for processing complex event query Q_4 is shown in Figure 4.4.

Q_4: EVENT SEQ(A x1, B x2, ¬ (C x3), D x4)
WHERE [attr1, attr2] ∧ x1.attr3 = '1' ∧ x1.attr4 > x4.attr4
WITHIN T

Meanwhile, SASE also proposes some related query optimization strategies, including Pushing Predicates Down (PPD) and Pushing Windows Down (PWD) to tackle the issues of huge intermediate results and sliding window constraints.

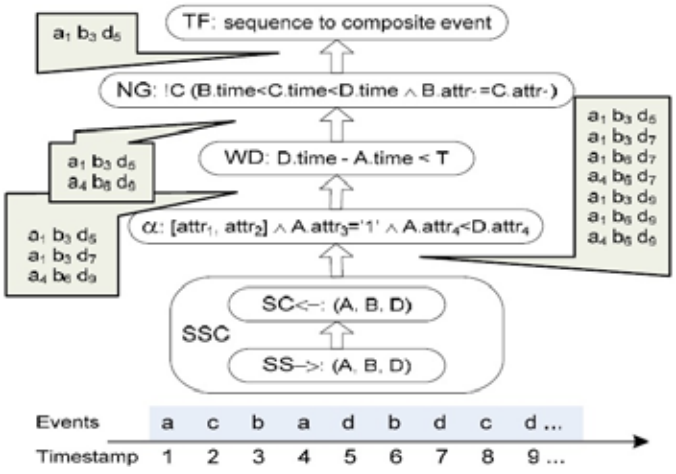


Figure 4.4. An Execution Plan for Query Q4.

The work in [43] tries to to improve the efficiency of complex RFID event detection with SASE framework where the domains of event attributes (e.g. tag ID) are quasi-infinite. Methods proposed include a delay matching method based on selectivity of injected events and two sliding window strategies based on time-slot and B+ tree. As an extension of SASE, SASE+ [70] employs an optimization strategy based on pattern match buffer sharing to support sharing among intermediate results, thus reduces the maintenance cost of intermediate results.

In RCEDA, to process RFID rules, the events from the rules are first constructed into an event graph, and then the event graph will be initialized as follows: i) propagate interval constraints in a top-down way; ii) assign event detection modes bottom-up based on event constructors and interval constraints; and assign pseudo event generation flags top-down based on the event detection modes. When pseudo events are created, they are placed in a sorted pseudo event queue based on their scheduled execution timestamps. The incoming RFID event queue is ordered based on observation timestamps. When the event is processed, the event engine fetches the earliest event from the two queues.

4. Advanced Topics on Complex Event Processing for Sensor Streams

In real world problems, complex event processing faces more challenges, such as the effect of event uncertainty and the disorder of events.

4.1 Probability of Events

Uncertainty of events is among the most important challenges of complex event detection, and there can be various reasons that produce probabilistic event data. Typical cases include conflicting readings, missed readings, or granularity mismatch.

Complex event detection on probabilistic data can be divided into two categories: local uncertainty detection and global uncertainty detection. When an tuple or object is independent of others, and the event detection only concerns with the uncertainty of itself, it is called ***local uncertainty detection***. On the other hand, when the event detection must consider the combined uncertainty among objects, it is called ***global uncertainty detection***. Generally, if the decision on whether an object satisfies a detection condition depends on other objects not involved in the same generation rule, global uncertainty has to be considered. Semantically, we have to examine the possible worlds one by one and count the probability that a combination of objects or tuples is an answer.

Probabilistic event processing has been studied in the context of query processing over probabilistic data streams. Jayram et al. [48] introduce a probabilistic stream model. Jayram et al. [48, 49] and Garofalakis [50] propose efficient algorithms for computing aggregate functions over uncertain data streams, where correlations across time are not considered. A hidden Markov model is used in [51] to support queries over probabilistic streams produced. The queries are limited to selections, projections, and aggregations. The method proposed in the Data Furnace project [53] extracts probabilistic events from imprecise sensor data. Its design relies on exploiting an inference engine to compute event probabilities, for example, using the work in [52].

Lahar[52] is an event processing system for probabilistic event streams. Lahar supports a much richer query model over probabilistic streams including sequences and joins. By exploiting the probabilistic nature of the data, Lahar yields a much higher recall and precision than deterministic techniques operating over only the most probable tuples.

4.2 Disorder of Events

The tuples in an event flow may be ordered or disordered on some attributes. When an order exists, some operations become easier and can be performed without the need of arbitrary storage; however, when this order is violated, it will be called "***event disorder***". Poset processing consists of performing operations on a set of tuples that may not be related by a total ordering. Any partially ordered set of tuples can

be processed in arbitrary ways within an event flow processing system by storing those tuples and retrieving as needed to match desired patterns. Most current research assumes events are ordered, and do not consider the concurrence and overlapping of events. However, in many real applications this assumption may not be valid.

Meanwhile, the real-time processing in temporal orders of event streams generated from distributed devices is a primary challenge for today's monitoring and tracking applications. In pervasive computing environments, event sequences might be out-of-order at the processing engine due to machine failure or network latency. Most systems [44, 45], either event-based or stream-based, assume a total ordering among event arrivals. Such existing technologies are likely to fail in such circumstances, either missing correct matches (i.e., false negatives) or producing incorrect matches (i.e., false positives). Supporting both in-order as well as out-of-order events efficiently and in real-time is an important research topic for complex event detection.

Based on the summary of different scenarios, the existing work on event disorder can be categorized into two types, one focusing on real time where the output is unordered, and another one focusing on the correctness where the output is ordered. If the input event stream to the query engine is unordered, it is reasonable to produce unordered output events. The method in [46] permits unordered sequence output based on an aggressive strategy. The aggressive strategy produces maximal output under the assumption that out-of-order event arrival is rare. In the case when out-of-order data arrival occurs, the results that have already been erroneously output will be corrected. One requirement here is that, for traditionally append-only streams, data cannot be updated once it is placed on a stream. Thus, a traditional append-only event model is no longer adequate. Another requirement is that, to enable the correction at any time, the access to historical operator states are needed until safe purging is possible. The upper bounds of K-slack could be used for periodic safe purging of the states of WinSeq and WinNeg operators when event instances are out of Window size K. This ensures that data are preserved so that any prior computation can be re-computed from its original input as needed. The approach extends the common append-only stream model to support the correction of prior released data on a stream. Two types of stream messages are used: Insertion tuple $< +, t >$ is induced by an out-of-order positive event, where t is a new sequence result. Deletion tuple $< -, t >$ is induced by an out-of-order negative event, such that t consists of the previously processed sequence. Deletion tuples cancel previous sequence results through the appearance of an

out-of-order negative event. Applications can thus distinguish between the types of tuples they receive.

If ordered output is needed, additional semantic information such as K-Slack factor or punctuation is needed to "unblock" the on-hold candidate sequences from being output. Two techniques are introduced to support this. A native approach [44, 45] on handling out-of-order event stream uses K-Slack as a priori bound on the out-of-order in the input streams. It buffers incoming events in the input queue for K time units until the ordering can be guaranteed. The major drawback of K-slack is the rigidity of the K parameter that cannot adapt to the variance in the network latencies existing in a heterogeneous RFID reader network. For example, one reasonable setting of K may be the maximum of the average latencies in the network. However, as the average latencies change, K may become either too large (thereby buffering unneeded data and introducing unnecessary inefficiencies and delays for the processing), or too small (thereby becoming inadequate for handling the out-of-order processing of the arriving events and resulting in inaccurate results). It also requires additional space and introduces more latency before allowing events being evaluated.

Another solution proposed to handle out-of-order data arrival is applying punctuation, namely, assertions inserted directly in the data stream confirming that, for instance, a certain value or time stamp will no longer appear in the future input streams [47, 46]. Permanent valid is achieved because results are only reported when they are known to be final. Relative small memory consumption is achieved by employing purging as early as possible. To safely purge data, meta-knowledge is needed to guarantee the nonoccurrence of future out-of-order data. A general method for meta-knowledge in streaming is to interleave dynamic constraints into the data streams, sometimes called punctuation. Based on this, a conservative method is proposed in [46]. It works under the assumption that out-of-order data may be common, and it produces output only when its correctness can be guaranteed. A partial order guarantee (POG) model is proposed to guarantee the correctness. Such techniques do require some services to be created first and appropriately inserting such assertions. Using POGs provides a simple and highly flexible solution. If the network latency were to fluctuate over time, it could be naturally captured by adjusting the POG generation without requiring any change of the query engine. Also, the query engine design can be agnostic to particularities of the domain or the environment. While it is conceivable that POGs themselves can arrive out-of-order, a punctuate operator could conservatively determine when POGs are released into the stream based on acknowledged receival of the events

in question. Hence, in practice, out-of-order POG may be delayed but would not arrive prematurely. Clearly, such delay or even complete loss of a POG would not cause any errors (such as incorrect purge of the operator state), rather it would in the worst case cause increased output latency. Fortunately, no incorrect results will be generated because the WinNeg operator would simply keep blocking until the subsequent POG arrives.

5. Conclusions and Summary

Sensor streams generated from sensor and RFID applications provide rich observations of physical objects. Event based processing of sensor data enables tracking and monitoring of physical objects and semantically interpreting complex event patterns. Event processing engines are essential to provide effective, efficient, and near real-time complex event processing of sensor data streams. Driven by the semantics of sensor events and event processing, and example use cases, we discuss two scenarios on event processing: event processing in sensor networks and event processing in RFID applications. For event processing in sensor networks, we present three major categories of approaches: statistical methods, topographical techniques, and edge detection algorithms. RFID events have their unique characteristics, and we discuss event specification languages, event detection models, event processing methods and their corresponding optimizations. Finally, we discuss two major challenges in practice, the effect of event uncertainty and the disorder of events.

References

[1] Subramaniam, S., Kalogeraki, V., Palpanas, T.: Distributed Real-time Detection and Tracking of Homogeneous Regions in Sensor Networks. RTSS, pp. 401–411. (2006)

[2] Liu, Y., Li, M.: Iso-map: Energy-efficient Contour Mapping in Wireless Sensor Networks. ICDCS, pp. 36–44. (2007)

[3] Li, M., Liu Y., Chen, L.: Non-threshold Based Event Detection for 3d Environment Monitoring in Sensor Networks. ICDCS. (2007)

[4] Li, M., Liu, Y.: Underground Structure Monitoring with Wireless Sensor Networks. IPSN. (2007)

[5] Chintalapudi, K. K., Govindan, R.: Localized Edge Detection in Sensor Fields. SNPA, pp. 59–70. (2003)

[6] Liu, J., Cheung, P., Zhao, F., Guibas, L.: A Dual-space Approach to Tracking and Sensor Management in Wireless Sensor Networks. WSNA, pp.131–139. (2002)

[7] Liu J., Zhao F., Cheung P., Guibas L.: Apply Geometric Duality to Energy Efficient non-local Phenomenon Awareness using Sensor Networks. IEEE Wireless Communication Magazine, pp. 62–68. (2004)

[8] Yang, H., Sikdar, B.: A Protocol for Tracking Mobile Targets using Sensor Networks. in SNPA, pp. 71–81. (2003)

[9] Zhang, W., Cao. G.: DCTC: Dynamic Convoy Tree-based Collaboration for Mobile Target Tracking. IEEE Transactions on Wireless Communications, vol. 3, No. 5, pp. 1689–1701. (2004)

[10] Lazos, L., Poovendran, R., Ritcey, J.A.: Probabilistic Detection of Mobile Targets in Heterogeneous Sensor Networks. IPSN, pp. 519–528. (2007)

[11] Jayant, G., Andreas, T., Randal, C., Alexander, S.: Model-Based Event Detection in Wireless Sensor Networks. CoRR abs/0901.3923. (2009)

[12] IBM. DB2 intelligent miner. available at `http://www-306.ibm.com/software/data/iminer/`. (2007)

[13] Deshpande, A., Madden, S.. Mauvedb: supporting model-based user views in database systems. ACM SIGMOD Conference. (2006)

[14] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M., Hong, W.: Model-driven Data Acquisition in Sensor Networks. VLDB Conference. (2004)

[15] Jain, A., Change, E., Wang, Y.: Adaptive Stream Resource Management Using Kalman Filters. ACM SIGMOD Conference. (2004)

[16] Chu, M., Haussecker, H., Zhao, F.: Scalable Information-driven Sensor Querying and Routing for Ad hoc Heterogeneous Sensor Networks. International Journal of High-Performance Computing Applications, vol. 16, no. 3. (2002)

[17] Grumbach, S., Rigaux, P., Segoufin, L.: Manipulating Interpolated Data is Easier than You Thought. VLDB Conference. (2000)

[18] Neugebauer, L.: Optimization and Evaluation of Database Queries including Embedded Interpolation Procedures. SIGMOD Conference. (1991)

[19] Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., Madden, S.: Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. IPSN, (2004)

[20] Gupta, H., Nacda, V., Das, S., Chowdhary, V.: Energy-efficient Gathering of Correlated Data in Sensor Networks. MobiHoc. (2005)

[21] Kotidis, Y.: Snapshot Queries: Towards Data-centric Sensor Networks. ICDE Conference. (2005)

[22] Deligiannakis, A., Kotidis, Y., and Roussopoulos, N.: Compressing Historical Information in Sensor Networks. SIGMOD Conference. (2004)

[23] Chu, D., Deshpande, A., Hellerstein, J., Hong, W.: Approximate Data Collection in Sensor Networks using Probabilistic Models. ICDE Conference. (2006)

[24] Silberstein, A., Braynard, R., Filpus, G., Puggioni, G., Gelfand, A., Munagala, K., Yang, J.: Data-driven Processing in Sensor Networks. Conference on Innovative Data Systems Research. (2007)

[25] Tulone, D., Madden, S.: PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks. European Conference on Wireless Sensor Networks. (2006)

[26] Ganeriwal, S., Srivastava, M.: Reputation-based Framework for High Integrity Sensor Networks. ACM Workshop on Security of Ad hoc and Sensor Networks (SASN), pp. 66–77. (2004)

[27] Hugh, T., Angela, Y,, Les, P., Frank, M.: Event-Driven Architecture. Addison-Wesley. (2009)

[28] Etzion, O., Niblett, P.: Event Processing in Action, Manning Publications. (2010)

[29] Govindan, R., Estrin, D.: Directed diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. International Conference on Mobile Computing and Networking. (2000)

[30] Halkidi, M., Kalogeraki, V., Gunopulos, D., Papadopoulos, D., Zeinalipour-Yazti, D., Vlachos, M.: Efficient online State Tracking using Sensor Networks. International Conference on Mobile Data Management (MDM). (2006)

[31] Olston, C., Loo, B.T., Widom, J.: Adaptive Precision Setting for Cached Approximate Values. SIGMOD Conference. (2001)

[32] Dutta, P., Grimmer, M., Arora, A., Bibyk, S., Culler, D.: Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. IPSN Conference. (2005)

[33] Cao, Q., Abdelzaher, T., He, T., Stankovic, J.: Towards Optimal Sleep Scheduling in Sensor Networks for Rare-event Detection. IPSN Conference. (2005)

[34] Girod, L., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H., Madden, S.: The Case for a Signal-Oriented Data Stream Management System. CIDR Conference. (2007)

[35] Abadi, D.J., Madden, S., Lindner, W.: Reed: Robust, Efficient Filtering and Event Detection in Sensor Networks. VLDB Conference. (2005)

[36] Madden, S., Franklin, M., Hellerstein, J., Hong, W.: Tinydb: an Acquisitional Query Processing System for Sensor Networks. ACM Trans. Database Syst. 30(1) (2005)

[37] Welsh, M.: Exposing Resource Tradeoffs in Region-based Communication Abstractions for Sensor Networks. SIGCOMM Comput. Commun. Rev. 34(1) (2004)

[38] Welsh, M., Mainland, G.: Programming Sensor Networks using Abstract Regions. Symposium on Networked Systems Design and Implementation (NSDI). (2004)

[39] Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and Beyond: New Aggregation Techniques for Sensor Networks. International Conference on Embedded Networked Sensor Systems. (2004)

[40] Sharifzadeh, M., Shahabi, C.: Supporting Spatial Aggregation in Sensor Network Databases. International Workshop on Geographic Information Systems (GIS). (2004)

[41] Chu, D., Tavakoli, A., Popa, L., Hellerstein, J.: Entirely Declarative Sensor Network Systems. VLDB Conference. (2006)

[42] Wu, E., Diao, Y., Rizvi, S.: High Performance Complex Event Processing over Streams. ACM SIGMOD Conference, pp. 407–418. (2006)

[43] Chen, Q., Li, Z. H., and Liu, H. L.: Optimizing Complex Event Processing over RFID Data Streams. ICDE Conference, pp. 1442–1444. (2008)

[44] Babu, S., et. al.: Exploiting K-constraints to Reduce Memory Overhead in Continuous Queries over Data Streams. ACM Transaction on Database Systems, Vol. 29, No. 3, pp. 545–580. (2004)

[45] Hammad, M. A. et.al.: Scheduling for Shared Window Joins over Data Streams. VLDB Conference, Vol.29, pp. 297–308. (2003)

[46] Liu, M., Li, M., Golovnya, D., Rundensteiner, E., Claypool, K.: Sequence Pattern Query Processing over Out-of-Order Event Streams. ICDE Conference, pp. 274–295. (2009)

[47] Ding, L., Mehta, N., Rundensteiner, E., Heineman, G.: Joining Punctuated Streams. EDBT Conference, pp. 587–604. (2004)

[48] Jayram, T., Kale, S., Vee, E.: Efficient Aggregation Algorithms for Probabilistic Data. SODA Conference. (2007)

[49] Jayram, T., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating Statistical Aggregates on Probabilistic Data Streams. PODS Conference. (2007)

[50] Cormode, G., Garofalakis, M.: Sketching Probabilistic Data Streams. SIGMOD Conference. (2007)

[51] Kanagal, B., Deshpande, A.: Online Filtering, Smoothing and Probabilistic Modeling of Streaming Data. ICDE Conference, pp. 1160–1169. (2008)

[52] Christopher, R., Julie, L., Magdalena, B., Dan, S.: Event Queries on Correlated Probabilistic Streams. ACM SIGMOD Conference, pp. 715–728. (2008)

[53] Garofalakis et. al.: Probabilistic Data Management for Pervasive Computing: The Data Furnace project. IEEE Data Engineering Bulletin, 29(1). (2006)

[54] Chakravarthy, S., et al.: *Composite events for active databases: Semantics, contexts and detection*. VLDB Conf., pp. 606–617. (1994)

[55] Zimmer, D., Unland, R.: *On the semantics of complex events in active database management systems*. ICDE Conf., pp. 392–399. (1999)

[56] Luckham, D.: *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Springer (2002).

[57] Bornhovd, C. et al.: *Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's AutoID Infrastructure*. VLDB Conference, pp. 1182–1188. (2004)

[58] Oracle. *Oracle Sensor Edge Server*. (2006) `http://www.oracle.com/technology/products/iaswe/edge_server`.

[59] IBM. *WebSphere RFID Premises Server*. (2004) `http://www-306.ibm.com/software/pervasive/ws_rfid_premises_server/`.

[60] Microsoft. *Microsoft's RFID 'Momentum' Includes Middleware Platform, Apps*. (2005) `http://www.eweek.com/c/a/Windows/Microsofts-RFID--Momentum-Includes-Middleware-Platform-Apps/`.

[61] UCLA. *UCLA WinRFID Middleware*. (2010) `http://www.wireless.ucla.edu/rfid/winrfid/Win-RFID.pdf`.

[62] Chandy, K. M., Schulte, R. W.: *Event Processing, Designing IT Systems for Agile Companies*. McGraw Hill. (2009)

[63] The participants of the Dagstuhl seminar on event processing *The Event Processing Manifesto*. (2010) `http://www.dagstuhl.de/10201`.

[64] Dayal, U., Buchmann, A. P., Chakravarthy, S.: *HiPAC Project Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann/Elsevier. (1996)

[65] Barga, R., et al.: *Consistent streaming through time: A vision for event stream processing*. CIDR Conference. (2007)

[66] Demers, A., et al.: *Cayuga: A general purpose event monitoring system*. CIDR Conference. pp, 412–422. (2007)

[67] Bass, T.: *Mythbusters: Event Stream Processing Versus Complex Event Processing*. DEBS. pp, 1–1. (2007)

[68] Gyllstrom D. et al.: *SASE: Complex Event Processing over Streams*. CIDR Conference, pp. 407–411. (2007)

[69] Gyllstrom, D. et al.: *On Supporting Kleene Closure over Event Streams*. ICDE Conference, pp. 1391–1393. (2008)

[70] Agrawal, J. et al.: *Efficient pattern matching over event streams*. ACM SIGMOD Conference, pp. 147–160, (2008).

[71] Liu, M. et al.: *Processing nested complex sequence pattern queries over event streams*. Data Management for Sensor Networks (DMSN), pp. 14–19. (2010)

[72] Chakravarthy, S., Mishra D.: *Snoop: an expressive event specification language for active databases*. Data Knowl. Eng. (DKE) 14(1), pp. 1–26, (1994).

[73] Wang, F. et al.: *Bridging physical and virtual worlds: Complex event processing for RFID data streams*. EDBT Conference, pp. 588–607. (2006)

[74] Wang, F. et al.: *Complex RFID event processing*. The VLDB Journal (2009) 18, pp. 913—931 (2009).

[75] Wang, F., Liu, P.: *Temporal Management of RFID Data*. VLDB Conference, pp. 1128–1139, (2005).

[76] Gonzalez, H., Han, J., Li, X. : *FlowCube: Constructuing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows*. VLDB Conference, pp. 834–845. (2006)

[77] Gehani, N. H. et al.: *Composite Event Specification in Active Databases: Model and Implementation*. VLDB Conference, pp. 327–338 (1992).

[78] Gatsiu, S., Dittrich, K. R.: *Events In an Active Object-oriented Database System*. International Conference on Rules in Database Systems, pp. 23–39. (1993)

[79] Jin, X., et al.: *Efficient Complex Event Processing over RFID Data Stream*. International Conference on Computer and Information Science, pp. 75–80 (2008).

[80] Hinze, A.: *Efficient Filtering of Composite Events*. British National Database Conference (BNCOD), pp. 207–225. (2003)

Chapter 5

DIMENSIONALITY REDUCTION AND FILTERING ON TIME SERIES SENSOR STREAMS

Spiros Papadimitriou
Rutgers University,
New Brunswick, NJ, USA
spapadim@business.rutgers.edu

Jimeng Sun
IBM Research,
Hawthorne, NY, USA
jimeng@us.ibm.com

Christos Faloutos
Carnegie Mellon University,
Pittsburgh, PA, USA
christos@cs.cmu.edu

Philip S. Yu
University of Illinois at Chicago,
Chicago, IL, USA
psyu@cs.uic.edu

Abstract This chapter surveys fundamental tools for dimensionality reduction and filtering of time series streams, illustrating what it takes to apply them efficiently and effectively to numerous problems. In particular, we show how least-squares based techniques (auto-regression and principal component analysis) can be successfully used to discover correlations both across streams, as well as across time. We also broadly overview work in the area of pattern discovery on time series streams, with applications in pattern discovery, dimensionality reduction, compression,

forecasting, and anomaly detection. We aim to provide a unified view of time series stream mining techniques for dimensionality reduction (analysis and data reduction across streams) and filtering (analysis and data reduction across time).

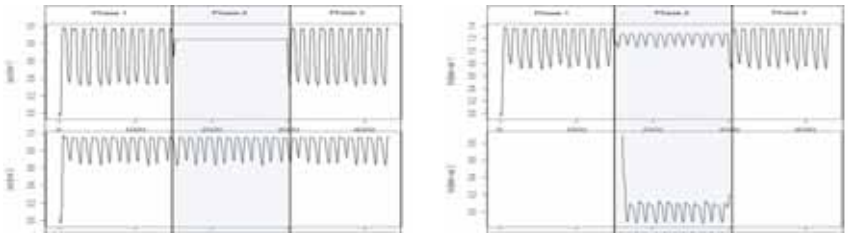
We describe methods that capture correlations and find hidden variables that describe trends in collections of streams. Discovered trends can then be used to quickly spot potential anomalies and do efficient forecasting. We describe a method which can incrementally find these correlation patterns and hidden variables, which summarize the key trends in the entire stream collection, with no buffering of stream values and without directly comparing pairs of streams. Moreover, it is any-time and dynamically detects changes. We also describe efficient online methods for quick forecasting (estimation of future values) and imputation (estimation of past, missing values) on multiple time series streams. Finally, we describe methods that can capture and summarize auto-correlations (correlations within a single series, across time), that also describe key trends. We also briefly explain how these techniques relate to others, and illustrate various trade-offs that are available to practitioners.

Keywords: streams, time series, filtering, dimensionality reduction, forecasting

1. Introduction

In this chapter, we consider the problem of capturing correlations both across multiple streams, as well as across time (auto-correlations). As we shall see, these two problems are inherently related, and similar techniques are applicable to both, even though the interpretation of the results may be different. In the first case, correlations across different streams allow us to find hidden variables that can summarize collections of time series data streams. In the second case, auto-correlations summarize patterns across time, that can capture regular or periodic trends in time series streams.

First we consider the case of correlations across many different streams. In general, we assume for simplicity that values from all streams are observed together; if that is not the case, then additional pre-processing or analysis may be necessary. Streams in a large collection are often inherently correlated (e.g., temperatures in the same building, traffic in the same network, prices in the same market, etc.) and it is possible to reduce hundreds of numerical streams into just a handful of *hidden variables* that compactly describe the key trends and dramatically reduce the complexity of further data processing. We will present an approach to do this incrementally.



(a) Sensor measurements (b) Hidden variables

Figure 5.1. Illustration of problem. Sensors measure chlorine in drinking water and show a daily, near sinusoidal periodicity during phases 1 and 3. During phase 2, some of the sensors are "stuck" due to a major leak. The extra hidden variable introduced during phase 2 captures the presence of a new trend. SPIRIT can also tell us which sensors participate in the new, "abnormal" trend (e.g., close to a construction site). In phase 3, everything returns to normal.

We describe a motivating scenario, to illustrate the problem we want to solve. Consider a large number of sensors measuring chlorine concentration in a drinkable water distribution network (see Figure 5.1, showing 15 days worth of data). Every five minutes, each sensor sends its measurement to a central node, which monitors and analyzes the streams in real time.

The patterns in chlorine concentration levels normally arise from water demand. If water is not refreshed in the pipes, existing chlorine reacts with pipe walls and micro-organisms and its concentration drops. However, if fresh water flows in at a particular location due to demand, chlorine concentration rises again. The rise depends primarily on how much chlorine is originally mixed at the reservoirs (and also, to a small extent, on the distance to the closest reservoir—as the distance increases, the peak concentration drops slightly, due to chemical reactions along the way). Thus, since demand typically follows a periodic pattern, chlorine concentration reflects that (see Figure 5.1a, bottom): it is high when demand is high and vice versa.

Assume that at some point in time, there is a major leak at some pipe in the network. Since fresh water flows in constantly (possibly mixed with debris from the leak), chlorine concentration at the nodes near the leak will be close to peak at all times.

Figure 5.1a shows measurements collected from two nodes, one away from the leak (bottom) and one close to the leak (top). At any time, a human operator would like to know how many trends (or *hidden variables*) are in the data and ask queries about them. Each hidden variable essentially corresponds to a group of correlated streams.

In this simple example, we first need to discover the correct number of hidden variables, which may change over time. Under normal operation, only one hidden variable is needed, which corresponds to the periodic pattern (Figure 5.1b, top). Both observed variables follow this hidden variable (multiplied by a constant factor, which is the *participation weight* of each observed variable into the particular hidden variable). Mathematically, the hidden variables are the *principal components* of the observed variables and the participation weights are the entries of the *principal direction* vectors (more precisely, this is true under certain assumptions, which will be explained later).

However, during the leak, a second trend is detected and a new hidden variable is introduced (Figure 5.1b, bottom). As soon as the leak is fixed, the number of hidden variables returns to one. If we examine the hidden variables, the interpretation is straightforward: The first one still reflects the periodic demand pattern in the sections of the network under normal operation. All nodes in this section of the network have a participation weight of ≈ 1 to the "periodic trend" hidden variable and ≈ 0 to the new one. The second hidden variable represents the additive effect of the catastrophic event, which is to cancel out the normal pattern. The nodes close to the leak have participation weights ≈ 0.5 to both hidden variables.

Summarizing this example, we find that (Figure 5.1): (i) Under normal operation (phases 1 and 3), there is one trend. The corresponding hidden variable follows a periodic pattern and all nodes participate in this trend. All is well. (ii) During the leak (phase 2), there is a *second* trend, trying to cancel the normal trend. The nodes with non-zero participation to the corresponding hidden variable can be immediately identified (e.g., they are close to a construction site). An abnormal event may have occurred in the vicinity of those nodes, which should be investigated.

Matters are further complicated when there are hundreds or thousands of nodes and more than one demand pattern. However, as we show later, it is still possible to extract the key trends from the stream collection, follow trend drifts and immediately detect outliers and abnormal events. Besides providing a concise summary of key trends/correlations among streams, correlations can be used to successfully deal with missing values and the discovered hidden variables can be used to do very efficient, resource-economic forecasting.

There are several other applications and domains in which correlation analysis and anomaly detection can be fruitfully combined. For example, (i) given more than 50,000 securities trading in US, on a second-by-second basis, detect patterns and correlations [62], (ii) given traffic

measurements [58], find routers that tend to go down together. In general, the discovered correlations and hidden variables have multiple uses. They provide a succinct summary to the user, they can help to do fast forecasting and detect outliers, and they facilitate interpolations and handling of missing values, as we discuss later.

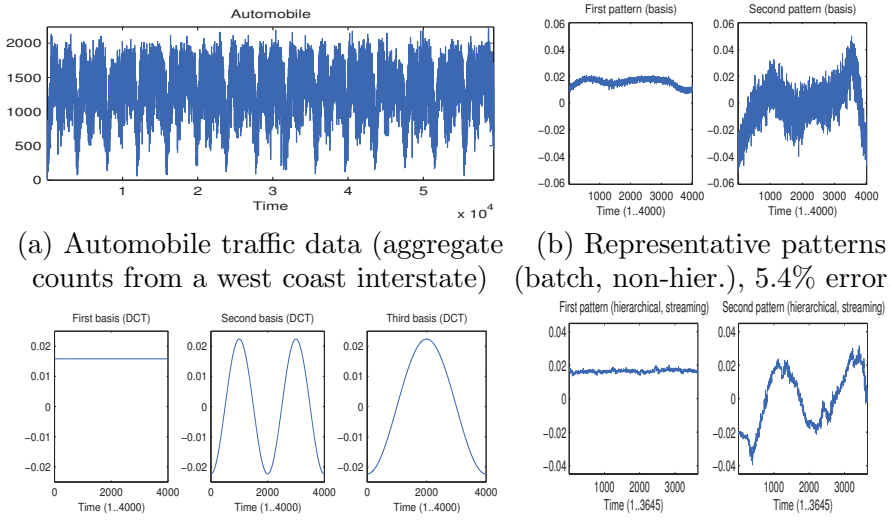
After giving an illustrative example where correlations across many streams arise, we consider the case of a *single* stream. Even in this case correlations are present. These correlations arise across values of the same stream at different times, instead across values from different streams. Values at different times are typically not independent, due to, for example, periodic or repeating patterns. These *auto*-correlations can be leveraged in similar ways, to perform dimensionality reduction (compression or filtering) across time. In fact, the problems of dimensionality reduction, filtering, and forecasting are closely related, as we shall see.

For purposes of illustration, consider the following example series in Figure 5.2a, which consists of automobile traffic counts in a large, west coast interstate. The data exhibit a clear daily periodicity. Also, in each day there is another distinct pattern of morning and afternoon rush hours. However, these peaks have distinctly different shapes: the morning one is more spread out, the evening one more concentrated and slightly sharper.

What we would ideally like to discover is: (i) The main trend in the data repeats at a window ("period") of approximately 4000 timestamps; (ii) A succinct "description" of that main trend that captures most of the recurrent information.

Figure 5.2b shows the output of a pattern discovery approach, based on filtering techniques very similar to those used for cross-stream correlations. These patterns indeed suggest that the "best" window is 4000 timestamps. Furthermore, the first pattern captures the average and the second pattern correctly captures the two peaks and also their approximate shape (the first one wide and the second narrower). For comparison, in Figure 5.2d shows the output of a fast, streaming computation scheme. In order to reduce the storage and computation requirements, our fast scheme tries to filter out some of the "noise" earlier, while retaining as many of the regularities as possible. However, which information should be discarded and which should be retained is once again decided based on the data *itself*. Thus, even though some information is unavoidably discarded, Figure 5.2b still correctly captures the main trends (average level, peaks and their shape).

For comparison, Figure 5.2c shows the best "local patterns" obtained using fixed bases. For illustration, we chose the Discrete Cosine Transform (DCT) on the first window of 4000 points. First, with the notable



(a) Automobile traffic data (aggregate counts from a west coast interstate)

(b) Representative patterns (batch, non-hier.), 5.4% error.

(c) Fixed bases (DCT) with highest coefficients. 7.7% error (8.4% with two coefficients).

(d) Representative patterns (streaming, hierarchical).

Figure 5.2. Automobile traffic, best selected window (about 1 day) and corresponding representative patterns.

exception of wavelets, most fixed-basis schemes cannot be easily used to capture information at arbitrary time scales. Also, any fixed-basis scheme (e.g., wavelets, Fourier, etc) would produce similar results which are heavily biased towards the shape of the *a priori* chosen bases or approximating functions. On the other hand, when bases are discovered from the data, we need additional storage space to explicitly represent them, which is not necessary when the bases are given.

In general, collections of semi-infinite, time-evolving streams can be modeled as values organized along several "dimensions[1]". One "dimension" corresponds to different streams in the collection. We first start by describing techniques that apply in this case. Time is another "dimension," that is somewhat special since it has an inherent ordering; we see how techniques for cross-stream analysis can be adapted for "cross-time" analysis.

We should emphasize that dimensionality reduction, filtering, and forecasting on time series data has been broadly studied in several disciplines. However, in this chapter we focus specifically on work in the context of data mining and knowledge discovery, with a special emphasis on streams and sensor data.

[1]Here "dimension" does not have the typical meaning in the linear algebraic sense.

Furthermore, as noted earlier, in this chapter we do not consider the cases where either inter-arrival times within a single stream vary wildly, or where arrival times across two different streams are not (approximately) synchronized. These settings have been studies somewhat less extensively, and are beyond our scope. Finally, it is possible to organize collections of streams into more than one "dimensions" (or modes), leading to tensor stream models [63]; this is also beyond the scope of this chapter.

The rest of the chapter is organized as follows: Section 2 presents work on data streams and stream mining, for both single and multiple time series streams. Section 3 and 4 overview some of the background of common models to characterize correlations across many series, as well as across time, respectively. Section 5 describes a method for efficient incremental update of multivariate forecasts, which can be used to spot unexpected values. Section 6 describes in detail a core method for anomaly detection based on these principles and Section 7 shows how its output can be interpreted and immediately utilized, both by humans, and for further data analysis. Section 8 illustrates the interplay between filtering and dimensionality reduction, showing how ideas related to Section 6 can be used for efficient and effective streaming pattern discovery across time, rather than across series. Finally, the conclusions are presented in Section 9.

2. Broader Overview

The area of dimensionality reduction and filtering is too extensive to be fully covered in a single chapter. Therefore, in this section, we will provide a broader overview of the related techniques, before going into some of the important techniques in greater detail. As mentioned before, in this chapter we focus specifically on work in the context of data mining and knowledge discovery, although correlation analysis has been both studied and used in numerous disciplines. Broadly speaking, the correlations can either be across streams (leading to dimensionality reduction) or across different time units in the same stream (leading to compression and filtering). Although this division is not perfect, as techniques are often related, we will next discuss these aspects of correlation analysis separately. Furthermore, for techniques that combine correlation analysis across streams and across time, interested readers may consult, e.g., [63].

2.1 Dimensionality reduction

Much of the work on stream mining has focused on finding interesting patterns in a single stream, but multiple streams have also attracted significant interest. Ganti et al. [22] propose a generic framework for stream mining. Guha et al. [25] propose a one-pass k-median clustering algorithm. [15] construct a decision tree online, by passing over the data only once. Later on, [29] and [54] addressed the problem of finding patterns over concept drifting streams.

The work in [35] propose parameter-free methods for classic data mining tasks (i.e., clustering, anomaly detection, classification), based on compression. The work in [36] proposes a multi-resolution clustering scheme for time series data. It uses the average coefficients (low frequencies) of the wavelet transform to perform k-means clustering and progressively refines the clusters by incorporating higher-level, detail coefficients. This approach requires much less time for convergence, compared to operating directly on the very high dimension of the original series. Both approaches require the complete data in advance. [4] propose a framework for *Phenomena Detection and Tracking (PDT)* in sensor networks. They define a phenomenon on discrete-valued streams and develop query execution techniques based on multi-way hash join with PDT-specific optimizations.

CluStream [1] is a flexible clustering framework with online and offline components. The online component extends micro-cluster information [61] by incorporating exponentially-sized sliding windows while coalescing micro-cluster summaries. Actual clusters are found by the offline component. StatStream [62] uses the DFT to summarise streams within a finite window and then compute the highest pairwise correlations among all pairs of streams, at each timestamp. BRAID [49] addresses the problem of discovering lag correlations among multiple streams. The focus is on time and space efficient methods for finding the earliest and highest peak in the cross-correlation functions between all pairs of streams. Similar to [42] (see below), BRAID employs a representation with fidelity that decreases with age. The work in [39] has studied how to efficiently compute pairwise correlations among large collections of time series, by combining compressed Fourier representations with graph partitioning techniques. Neither CluStream, StatStream, or BRAID explicitly focus on discovering hidden variables.

MUSCLES [58] is exactly designed to do forecasting (thus it could handle missing values). However, it can not find hidden variables and it scales poorly for a large number of streams n, since it requires at

least quadratic space and time, or expensive reorganization (*selective MUSCLES*).

The problem of *principal components analysis (PCA)* and SVD on streams has been addressed in [44] and [24]. Both of these approaches focus on discovering linear correlations among multiple streams and on applying these correlations for further data processing and anomaly detection [44]. [24] first does dimensionality reduction with random projections, and then periodically computes the SVD. However, the method incurs some overhead because of the SVD re-computation and it can not easily handle missing values. Also related is the work of [13] which uses a different formulation of linear correlations and focuses on compressing historical data, mainly for power conservation in sensor networks. Finally, the work in [6] proposes an approach to combine segmentation of multidimensional series with dimensionality reduction. The reduction is on the segment representatives and it is performed across dimensions (similar to [44]), not along time, and the approach is not applicable to streams.

Beyond discovering and leveraging possibly evolving patterns in streaming series in an unsupervised fashion, the work in [55] leverages commonalities in a set of given query patterns, in order to discover them efficiently among streaming data. The work in [53] and [50] studies "anytime" algorithms for nearest-neighbor classification on streams of either single items or batches of items. In such a setting, available resources (time or buffer space) can be traded-off for increased accuracy. The work in [5] develops anytime algorithms for outlier detection on data streams, based on a hierarchical cluster representation as a reduced representation of the incoming data.

Closely related to [48] (see below) is [21], which develops a joint compression framework for collections of time series, while providing guarantees on maximum reconstruction error, as well as also allowing queries to be answered using indices directly on the compressed representation.

Sensor streams. A number of related techniques for correlation and prediction across multiple sensor streams are covered in [2] [57] [11] [51]. Such methods can be used in order to improve the power efficiency of a sensor network, because only the non-redundant sensors need to transmit their data at higher sampling rates.

2.2 Compression and filtering

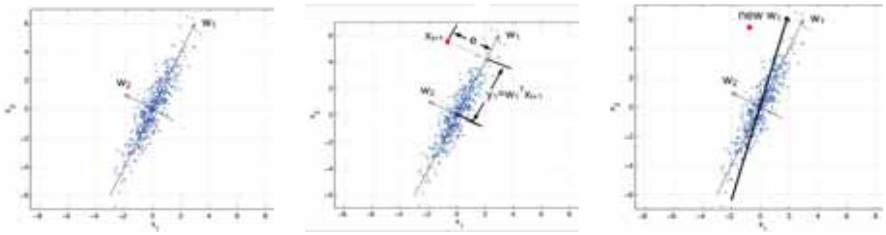
Initial work on time series representation [3, 19] uses the Fourier transform. Even more recent work uses fixed, predetermined bases or approx-

imating functions. APCA [8] and other similar approaches approximate the time series with piecewise constant or linear functions. DAWA [28] combines the DCT and DWT. However, all these approaches focus on compressing the time series for indexing purposes, and not on pattern discovery. AWSOM [43] first applies the wavelet transform. As the authors observe, just a few wavelet coefficients do not capture all patterns in practice, so AWSOM subsequently captures trends by fitting a linear auto-regressive model at each time scale.

The seminal work of [12] for rule discovery in time series is based on sequential patterns extracted after a discretization step. Other work has also focused on finding *representative trends* [32]. A representative trend is a subsequence of the time series that has the smallest sum of distances from all other subsequences of the same length. The proposed method employs random projections and FFT to quickly compute the sum of distances. This does not apply directly to streams and it is not easy to extend, since each section has to be compared to all others. Our approach is complementary and could conceivably be used in place of the FFT in this setting. Related to representative trends are *motifs* [45, 10]. Intuitively, these are frequently repeated subsequences, i.e., subsequences of a given length which match (in terms of some distance and a given distance threshold) a large number of other subsequences of the same time series. More recently, vector quantization has been used for time series compression [38, 37]. The first focuses on finding good-quality and intuitive distance measures for indexing and similarity search and is not applicable to streams, while the second focuses on reducing power consumption for wireless sensors. Finally, other work on stream mining includes approaches for periodicity [18] and periodic patterns [17] discovery.

More recently, [48] have studied how efficiently store time series, while allowing computation of several quantities (e.g., correlations, histograms) directly in the compressed domain, by leveraging multi-scale analysis to obtain sparse time/frequency representations of time series. The work of [47] considers the problem of discovering patterns on a single time series by clustering series from one time series stream, and proposes an MDL-based framework for efficiently discovering good clusters.

Approaches for regression on time series and streams include [9] and *amnesic functions* [42]. Both of these estimate the best fit of a given function (e.g., linear or low-degree polynomial), they work by merging the estimated fit on consecutive windows and can incorporate exponential-size time windows placing less emphasis on the past. However, both of these approaches employ a fixed, given set of approximating functions.



(a) Original $\mathbf{w}_1$ (b) Update process (c) Resulting $\mathbf{w}_1$

Figure 5.3. Illustration of updating $\mathbf{w}_1$ when a new point $\mathbf{x}_{t+1}$ arrives.

Our approach might better be described as agnostic, rather than amnesic.

A very recent and interesting application of the same principles is on correlation analysis of complex time series through change-point scores [31]. Finally, related ideas have been used in other fields, such as in image processing for image denoising [40, 30] and physics/climatology for nonlinear prediction in phase space [59]. However, none of these approaches address incremental computation in streams. More generally, the potential of this general approach has not received attention in time series and stream processing literature. We demonstrate that its power can be harnessed at very small cost, no more than that of the widely used wavelet transform.

The recently developed theory of *compressed sensing* (e.g., [16] and [26]) studies the problem of signal summarization and reconstruction based on observation of a subset of its values. More precisely, this work develops a framework for estimating the projections of a signal into a given set of basis functions from a small set of samples of its values.

3. Principal Component Analysis (PCA)

Here we give a brief overview of PCA [33], explaining the main intuition. We use standard matrix algebra notation: vectors are lower-case bold, matrices are upper-case bold, and scalars are in plain font. The transpose of matrix $\mathbf{X}$ is denoted by $\mathbf{X}^T$. In the following, $\mathbf{x}_t \equiv [x_{t,1}\, x_{t,2} \cdots x_{t,n}]^T \in \mathbb{R}^n$ is the column-vector. of stream values at time t. We adhere to the common convention of using column vectors and writing them out in transposed form. The stream data can be viewed as a continuously growing $t \times n$ matrix $\mathbf{X}_t := [\mathbf{x}_1\, \mathbf{x}_2 \cdots \mathbf{x}_t]^T \in \mathbb{R}^{t \times n}$, where one new row is added at each time tick t. In the chlorine example, $\mathbf{x}_t$ is the measurements column-vector at t over all the sensors, where n is the number of chlorine sensors and t is the measurement time-stamp.

Typically, in collections of n-dimensional points $\mathbf{x}_t \equiv [x_{t,1} \ldots, x_{t,n}]^T$, $t = 1, 2, \ldots$, there exist correlations between the n dimensions (which correspond to streams in our setting). These can be captured by principal components analysis (PCA). Consider for example the setting in Figure 5.3. There is a visible linear correlation. Thus, if we represent every point with its projection on the direction of $\mathbf{w}_1$, the error of this approximation is very small. In fact, the first principal direction $\mathbf{w}_1$, is the *optimal* in the following sense.

DEFINITION 5.1 (FIRST PRINCIPAL COMPONENT) *Given a collection of n-dimensional vectors $\mathbf{x}_\tau \in \mathbb{R}^n$, $\tau = 1, 2, \ldots, t$, the* first principal direction $\mathbf{w}_1 \in \mathbb{R}^n$ *is the vector minimizing the sum of squared residuals, i.e.,*

$$\mathbf{w}_1 := \arg\min_{\|w\|=1} \sum_{\tau=1}^{t} \|\mathbf{x}_\tau - (\mathbf{w}\mathbf{w}^T)\mathbf{x}_\tau\|^2.$$

The projection of $\mathbf{x}_\tau$ on $\mathbf{w}_1$ is the first principal component (PC) $y_{\tau,1} := \mathbf{w}_1^T \mathbf{x}_\tau$, $\tau = 1, \ldots, t$.

Note that, since $\|\mathbf{w}_1\| = 1$, we have $(\mathbf{w}_1\mathbf{w}_1^T)\mathbf{x}_\tau = (\mathbf{w}_1^T\mathbf{x}_\tau)\mathbf{w}_1 = y_{\tau,1}\mathbf{w}_1 =: \tilde{\mathbf{x}}_\tau$, where $\tilde{\mathbf{x}}_\tau$ is the projection of $y_{\tau,1}$ back into the original n-D space. That is, $\tilde{\mathbf{x}}_\tau$ is the *reconstruction* of the original measurements from the first PC $y_{\tau,1}$. More generally, PCA will produce k vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k$ such that, if we represent each n-D data point $\mathbf{x}_t := [x_{t,1} \cdots x_{t,n}]$ with its k-D projection $\mathbf{y}_t := [\mathbf{w}_1^T\mathbf{x}_t \cdots \mathbf{w}_k^T\mathbf{x}_t]^T$, then this representation minimises the squared error $\sum_\tau \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2$. Furthermore, the principal directions are orthogonal, so the principal components $y_{\tau,i}, 1 \le i \le k$ are *by construction uncorrelated*, i.e., if $\mathbf{y}^{(i)} := [y_{1,i}, \ldots, y_{t,i}, \ldots]^T$ is the stream of the i-th principal component, then $\left(\mathbf{y}^{(i)}\right)^T \mathbf{y}^{(j)} = 0$ if $i \neq j$.

OBSERVATION 3.1 (DIMENSIONALITY REDUCTION) *If we represent each n-dimensional point $\mathbf{x}_\tau \in \mathbb{R}^n$ using all n principal components, then the error $\|\mathbf{x}_\tau - \tilde{\mathbf{x}}_\tau\| = 0$. However, in typical datasets, we can achieve a very small error using only k principal components, where $k \ll n$.*

In the context of the chlorine example, each point in Figure 5.3 would correspond to the 2-D projection of $\mathbf{x}_\tau$ (where $1 \le \tau \le t$) onto the first two principal directions, $\mathbf{w}_1$ and $\mathbf{w}_2$, which are the most important according to the distribution of $\{\mathbf{x}_\tau \mid 1 \le \tau \le t\}$. The principal components $y_{\tau,1}$ and $y_{\tau,2}$ are the coordinates of these projections in the orthogonal coordinate system defined by $\mathbf{w}_1$ and $\mathbf{w}_2$.

However, batch methods for estimating the principal components require time that depends on the duration t, which grows to infinity. In fact, the principal directions are the eigenvectors of $\mathbf{X}_t^T\mathbf{X}_t$, which are

Table 5.1. Description of notation.

Symbol	Description
$\mathbf{x}, \ldots$	Column vectors (lowercase boldface).
$\mathbf{A}, \ldots$	Matrices (uppercase boldface).
$\mathbf{x}_t$	The n stream values $\mathbf{x}_t := [x_{t,1} \cdots x_{t,n}]^T$ at time t.
n	Number of streams.
$\mathbf{w}_i$	The i-th participation weight vector (i.e., principal direction).
k	Number of hidden variables.
$\mathbf{y}_t$	Vector of hidden variables (i.e., principal components) for $\mathbf{x}_t$, i.e., $\mathbf{y}_t \equiv [y_{t,1} \cdots y_{t,k}]^T := [\mathbf{w}_1^T \mathbf{x}_t \cdots \mathbf{w}_k^T \mathbf{x}_t]^T$.
$\tilde{\mathbf{x}}_t$	Reconstruction of $\mathbf{x}_t$ from the k hidden variable values, i.e., $\tilde{\mathbf{x}}_t := y_{t,1}\mathbf{w}_1 + \cdots + y_{t,k}\mathbf{w}_k$.
E_t	Total energy up to time t.
$\tilde{E}_{t,i}$	Total energy captured by the i-th hidden variable, up to time t.
f_E, F_E	Lower and upper bounds on the fraction of energy we wish to maintain via SPIRIT's approximation.

best computed through the singular value decomposition (SVD) of $\mathbf{X}_t$. Space requirements also depend on t. Clearly, in a stream setting, it is impossible to perform this computation at every step, aside from the fact that we don't have the space to store all past values. We will address this problem in Section 6, where we present a solution that works without buffering *any* past values.

4. Auto-Regressive Models and Recursive Least Squares

In this section we review some of the background on popular forecasting methods for time series.

4.1 Auto-Regressive (AR) Modeling

Auto-regressive models are the most widely known and used—more information can be found in, e.g., [7]. The main idea is to express x_t as a function of its previous values, plus (filtered) noise ϵ_t:

$$x_t \;=\; \phi_1 x_{t-1} \;+\; \ldots \;+\; \phi_W x_{t-W} \;+\; \epsilon_t, \tag{5.1}$$

where W is a the forecasting window size. Seasonal variants (SAR, SAR(I)MA) also use window offsets that are multiples of a single, fixed period (i.e., besides terms of the form y_{t-i}, the equation contains terms of the form y_{t-Si} where S is a constant).

If we have a collection of n time series $x_{t,i}$, $1 \leq i \leq n$ then multivariate AR simply expresses $x_{t,i}$ as a linear combination of previous values of

all streams (plus noise), i.e.,

$$\begin{aligned} x_{t,i} \;=\; & \phi_{1,1}x_{t-1,1} \;+\; \ldots \;+\; \phi_{1,W}x_{t-W,1} \;+ \\ & \ldots + \\ & \phi_{n,1}x_{t-1,n} \;+\; \ldots \;+\; \phi_{n,W}x_{t-W,n} \;+\; \epsilon_t. \end{aligned} \tag{5.2}$$

4.2 Recursive Least Squares (RLS)

Recursive Least Squares (*RLS*) is a method that allows dynamic update of a least-squares fit. The least squares solution to an overdetermined system of equations $\mathbf{Xb} = \mathbf{y}$ where $\mathbf{X} \in \mathbb{R}^{m \times k}$ (measurements), $\mathbf{y} \in \mathbb{R}^m$ (output variables) and $\mathbf{b} \in \mathbb{R}^k$ (regression coefficients to be estimated) is given by the solution of $\mathbf{X}^T\mathbf{Xb} = \mathbf{X}^T\mathbf{y}$. Thus, all we need for the solution are the projections

$$\mathbf{P} \equiv \mathbf{X}^T\mathbf{X} \qquad \text{and} \qquad \mathbf{q} \equiv \mathbf{X}^T\mathbf{y}$$

We need only space $O(k^2 + k) = O(k^2)$ to keep the model up to date. When a new row $\mathbf{x}_{m+1} \in \mathbb{R}^k$ and output y_{m+1} arrive, we can update

$$\begin{aligned} \mathbf{P} &\leftarrow \mathbf{P} + \mathbf{x}_{m+1}\mathbf{x}_{m+1}^T \quad \text{and} \\ \mathbf{q} &\leftarrow \mathbf{q} + y_{m+1}\mathbf{x}_{m+1}. \end{aligned}$$

In fact, it is possible to update the regression coefficient vector $\mathbf{b}$ without explicitly inverting $\mathbf{P}$ to solve $P\mathbf{b} = \mathbf{P}^{-1}\mathbf{q}$. In particular (see, e.g., [60]) the update equations are

$$\mathbf{G} \leftarrow \mathbf{G} - (1 + \mathbf{x}_{m+1}^T\mathbf{G}\mathbf{x}_{m+1})^{-1}\mathbf{G}\mathbf{x}_{m+1}\mathbf{x}_{m+1}^T\mathbf{G} \tag{5.3}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \mathbf{G}\mathbf{x}_{m+1}(\mathbf{x}_{m+1}^T\mathbf{b} - y_{m+1}), \tag{5.4}$$

where the matrix $\mathbf{G}$ can be initialized to $\mathbf{G} \leftarrow \epsilon\mathbf{I}$, with ϵ a small positive number and $\mathbf{I}$ the $k \times k$ identity matrix.

RLS and AR In the context of auto-regressive modeling (Eq. 5.1), we have one equation for each stream value $x_{w+1}, \ldots, x_t, \ldots$, i.e., the m-th row of the $\mathbf{X}$ matrix above is

$$\mathbf{X}_m = [x_{m-1}\ x_{m-2}\ \cdots\ x_{m-w}]^T \in \mathbb{R}^w$$

and $z_m = x_m$, for $t - w = m = 1, 2, \ldots$ $(t > w)$. In this case, the solution vector $\mathbf{b}$ consists precisely of the auto-regression coefficients in Eq. 5.1, i.e.,

$$\mathbf{b} = [\phi_1\ \phi_2\ \cdots\ \phi_w]^T \in \mathbb{R}^w.$$

RLS can be similarly used for multivariate AR model estimation.

5. MUSCLES

MUSCLES (MUlti-SequenCe LEast Squares) [58] tries to predict the value of one stream, $x_{t,i}$ based on the previous values from all streams, $x_{t-l,j}$, $l > 1$, $1 \leq j \leq n$ and current values from other streams, $x_{t,j}$, $j \neq i$. It uses multivariate autoregression, thus the prediction $\hat{x}_{t,i}$ for a given stream i is, similar to Eq. 5.2

$$\begin{aligned}\hat{x}_{t,i} \;=\; & \phi_{1,0}x_{t,1} + \phi_{1,1}x_{t-1,1} \;+\; \ldots \;+\; \phi_{1,W}x_{t-W,1} \;+ \\ & \ldots \;+ \\ & \phi_{i-1,0}x_{t-1,i-1} \;+\; \phi_{i-1,1}x_{t-1,i-1} \;+\; \ldots \;+\; \phi_{i-1,w}x_{t-W,i-1} \;+ \\ & \qquad\qquad \phi_{i,1}x_{t-1,i} \;+\; \ldots \;+\; \phi_{i,w}x_{t-W,i} \;+ \\ & \phi i+1,0x_{t,i+1} \;+\; \phi_{i+1,1}x_{t-1,i+1} \;+\; \ldots \;+\; \phi_{i+1,w}x_{t-W,i+1} \;+ \\ & \ldots + \\ & \phi_{n,0}x_{t,n} \;+\; \phi_{n,1}x_{t-1,n} \;+\; \ldots \;+\; \phi_{n,W}x_{t-W,n} \;+\; \epsilon_t.\end{aligned}$$

and employs RLS to continuously update the coefficients $\phi_{i,j}$ such that the prediction error

$$\sum_{\tau=1}^{t}(\hat{x}_{\tau,i} - x_{\tau,i})^2$$

is minimized. Note that the above equation has one dependent variable (the estimate $\hat{x}_{t,i}$) and $v = W * n + n - 1$ independent variables (the past values of all streams plus the current values of all other streams except i).

Exponentially forgetting MUSCLES employs a forgetting factor $0 < \lambda \leq 1$ and minimizes instead

$$\sum_{\tau=1}^{t}\lambda^{t-\tau}(\hat{x}_{\tau,i} - x_{\tau,i})^2.$$

For $\lambda < 1$, errors for old values are down-weighted by an exponential factor, hence permitting the estimate to adapt as sequence characteristics change.

5.1 Selective MUSCLES

In case we have too many time sequences (e.g., $n = 100,000$ nodes in a network, producing information about their load every minute), even the incremental version of MUSCLES will suffer. The solution to this problem is based on the conjecture that we do not really need information from every sequence to make a good estimation of a missing value. Much of the benefit of using multiple sequences may be captured

by using only a small number of carefully selected other sequences. We can thus do some preprocessing of a training set, to find a promising subset of sequences, and to apply MUSCLES only to those (hence the name Selective MUSCLES).

Assume that sequence i is the one notoriously delayed and we need to estimate its "delayed" values $x_{t,i}$. For a given tracking window span W, among the $v = W * n + n - 1$ independent variables, we have to choose the ones that are most useful in estimating the delayed value of $x_{t,i}$. More generally, we want to solve the following

PROBLEM 5.1 (SUBSET SELECTION) *Given v independent variables $x_1, x_2, \ldots, x_v$ and a dependent variable y with N samples each, find the best b ($< v$) independent variables to minimize the mean-square error for $\hat{y}$ for the given samples.*

We need a measure of goodness to decide which subset of b variables is the best we can choose. Ideally, we should choose the best subset that yields the smallest estimation error in the future. Since, however, we don't have future samples, we can only infer the *expected estimation error* (EEE for short) from the available samples as follows:

$$\text{EEE}(\mathcal{S}) = \sum_{t=1}^{N} (y[t] - \hat{y}_{\mathcal{S}}[t])^2$$

where $\mathcal{S}$ is the selected subset of variables and $\hat{y}_{\mathcal{S}}[t]$ is the estimation based on $\mathcal{S}$ for the t-th sample. Note that, thanks to Eq. 5.3, EEE($\mathcal{S}$) can be computed in $O(N \cdot \|\mathcal{S}\|^2)$ time. Let's say that we are allowed to keep only $b = 1$ independent variable. Which one should we choose? Intuitively, we could try the one that has the highest (in absolute value) correlation coefficient with y. It turns out that this is indeed optimal: (to satisfy the unit variance assumption, we will normalize samples by the sample variance within the window.)

LEMMA 5.2 *Given a dependent variable y, and v independent variables with unit variance, the best single variable to keep to minimize* EEE($\mathcal{S}$) *is the one with the highest absolute correlation coefficient with y.*

Proof. For a single variable, if a is the least squares solution, we can express the error in matrix form as

$$\text{EEE}(\{x_i\}) = \|\mathbf{y}\|^2 - 2a(\mathbf{y}^T \mathbf{x}_i) + a^2 \|\mathbf{x}_i\|^2.$$

Let d and p denote $\|\mathbf{x}_i\|^2$ and $(\mathbf{x}T\mathbf{y})$, respectively. Since $a = d^{-1}p$, EEE($\{x_i\}$) $= \|y\|^2 - p^2 d^{-1}$. To minimize the error, we must choose x_i

which maximize p^2 and minimize d. Assuming unit-variance ($d = 1$), such x_i is the one with the biggest correlation coefficient to y. This concludes the proof. □

The question is how we should handle the case when $b > 1$. Normally, we should consider all the possible groups of b independent variables, and try to pick the best. This approach explodes combinatorially; thus we propose to use a greedy algorithm. At each step s, we select the independent variable x_s that minimizes the EEE for the dependent variable y, in light of the $s-1$ independent variables that we have already chosen in the previous steps.

The bottleneck of the algorithm is clearly the computation of EEE. Since it computes EEE approximately $O(v \cdot b)$ times and each computation of EEE requires $O(N \cdot b^2)$ in average, the overall complexity mounts to $O(N \cdot v \cdot b^3)$. To reduce the overhead, we observe that intermediate results produced for EEE($\mathcal{S}$) can be re-used for EEE($\mathcal{S} \cup \{x\}$).

LEMMA 5.3 *The complexity of the greedy selection algorithm is* $O(N \cdot v \cdot b^2)$.

Proof. Let $\mathcal{S}^+$ be $\mathcal{S} \cup \{x\}$. The core in computing EEE($\mathcal{S}^+$) is the inverse of $\mathbf{D}_{\mathcal{S}^+} = (\mathbf{X}_{\mathcal{S}^+}^T \mathbf{X}_{\mathcal{S}^+})$. Thanks to block matrix inversion formula [34] (p. 656) and the availability of $D_{\mathcal{S}}^{-1}$ from the previous iteration step, it can be computed in $O(N \cdot |\mathcal{S}| + |S|^2)$. Hence, summing it up over $v - |\mathcal{S}|$ remaining variables for each b iteration, we have $O(N \cdot v \cdot b^2 + v \cdot b^3)$ complexity. Since $N \gg b$, it reduces to $O(N \cdot v \cdot b^2)$. □

Subset-selection can be done infrequently and off-line, say every $N = W$ time-ticks. The optimal choice of the reorganization window is beyond the scope of this chapter. Potential solutions include (a) doing reorganization during off-peak hours, (b) triggering a reorganization whenever the estimation error for by increases above an application-dependent threshold etc. Also, by normalizing the training set, the unit-variance assumption in Theorem 1 can be easily satisfied.

6. Tracking Correlations and Hidden Variables: SPIRIT

In this section we present a framework for discovering patterns in multiple streams. In the next section, we show how these can be used to perform effective, low-cost forecasting. We use auto-regression for its simplicity, but our framework allows any forecasting algorithm to take advantage of the compact representation of the stream collection.

Problem definition Given a collection of n co-evolving, semi-infinite streams, producing a value $x_{t,j}$, for every stream $1 \leq j \leq n$ and for every time-tick $t = 1, 2, \ldots$, SPIRIT does the following: (i) Adapts the number k of *hidden variables* necessary to explain/summarise the main trends in the collection. (ii) Adapts the *participation weights* $w_{i,j}$ of the j-th stream on the i-th hidden variable ($1 \leq j \leq n$ and $1 \leq i \leq k$), so as to produce an accurate summary of the stream collection. (iii) Monitors the hidden variables $y_{t,i}$, for $1 \leq i \leq k$. (iv) Keeps updating all the above efficiently.

More precisely, SPIRIT operates on the column-vectors of observed stream values $\mathbf{x}_t \equiv [x_{t,1}, \ldots, x_{t,n}]^T$ and continually updates the participation weights $w_{i,j}$. The *participation weight vector* $\mathbf{w}_i$ for the i-th principal direction is $\mathbf{w}_i := [w_{i,1} \cdots w_{i,n}]^T$. The hidden variables $\mathbf{y}_t \equiv [y_{t,1}, \ldots, y_{t,k}]^T$ are the projections of $\mathbf{x}_t$ onto each $\mathbf{w}_i$, over time (see Table 5.1), i.e.,

$$y_{t,i} := w_{i,1}x_{t,1} + w_{i,2}x_{t,2} + \cdots + w_{i,n}x_{t,n},$$

SPIRIT also adapts the number k of hidden variables necessary to capture most of the information. The adaptation is performed so that the approximation achieves a desired mean-square error. In particular, let $\tilde{\mathbf{x}}_t = [\tilde{x}_{t,1} \cdots \tilde{x}_{t,n}]^T$ be the *reconstruction* of $\mathbf{x}_t$, based on the weights and hidden variables, defined by

$$\tilde{x}_{t,j} := w_{1,j}y_{t,1} + w_{2,j}y_{t,2} + \cdots + w_{k,j}y_{t,k},$$

or more succinctly, $\tilde{\mathbf{x}}_t = \sum_{i=1}^{k} y_{i,t}\mathbf{w}_i$.

In the chlorine example, $\mathbf{x}_t$ is the n-dimensional column-vector of the original sensor measurements and $\mathbf{y}_t$ is the hidden variable column-vector, both at time t. The dimension of $\mathbf{y}_t$ is 1 before/after the leak ($t < 1500$ or $t > 3000$) and 2 during the leak ($1500 \leq t \leq 3000$), as shown in Figure 5.1.

Definition 5.4 (SPIRIT Tracking) *SPIRIT updates the participation weights $w_{i,j}$ so as to guarantee that the reconstruction error $\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2$ over time is predictably small.*

This informal definition describes what SPIRIT does. The precise criteria regarding the reconstruction error will be explained later. If we assume that the $\mathbf{x}_t$ are drawn according to some distribution that does not change over time (i.e., under *stationarity* assumptions), then the weight vectors $\mathbf{w}_i$ converge to the principal directions. However, even if there are non-stationarities in the data (i.e., gradual drift), in practice we can deal with these very effectively, as we explain later.

An additional complication is that we often have missing values, for several reasons: either failure of the system, or delayed arrival of some measurements. For example, the sensor network may get overloaded and fail to report some of the chlorine measurements in time or some sensor may temporarily black-out. At the very least, we want to continue processing the rest of the measurements.

6.1 Tracking the Hidden Variables

The first step is, for a given k, to incrementally update the k participation weight vectors $\mathbf{w}_i$, $1 \leq i \leq k$, so as to summarise the original streams with only a few numbers (the hidden variables). In Section 6.2, we describe the complete method, which also adapts k.

For the moment, assume that the number of hidden variables k is given. Furthermore, our goal is to minimise the average reconstruction error $\sum_t \|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2$. In this case, the desired weight vectors $\mathbf{w}_i, 1 \leq i \leq k$ are the principal directions and it turns out that we can estimate them incrementally.

We use an algorithm based on adaptive filtering techniques [56, 27], which have been tried and tested in practice, performing well in a variety of settings and applications (e.g., image compression and signal tracking for antenna arrays). We experimented with several alternatives [41, 14] and found this particular method to have the best properties for our setting: it is very efficient in terms of computational and memory requirements, while converging quickly, with no special parameters to tune. The main idea behind the algorithm is to read in the new values $\mathbf{x}_{t+1} \equiv [x_{(t+1),1}, \ldots, x_{(t+1),n}]^T$ from the n streams at time $t+1$, and perform three steps:

1 Compute the hidden variables $y'_{t+1,i}, 1 \leq i \leq k$, based on the *current* weights $\mathbf{w}_i, 1 \leq i \leq k$, by projecting $\mathbf{x}_{t+1}$ onto these.

2 Estimate the reconstruction error ($\mathbf{e}_i$ below) and the energy, based on the $y'_{t+1,i}$ values.

3 Update the estimates of $\mathbf{w}_i, 1 \leq i \leq k$ and output the *actual* hidden variables $y_{t+1,i}$ for time $t+1$.

To illustrate this, Figure 5.3b shows the $\mathbf{e}_1$ and y_1 when the new data $\mathbf{x}_{t+1}$ enter the system. Intuitively, the goal is to adaptively update $\mathbf{w}_i$ so that it quickly converges to the "truth." In particular, we want to update $\mathbf{w}_i$ more when $\mathbf{e}_i$ is large. However, the magnitude of the update should also take into account the past data currently "captured" by $\mathbf{w}_i$. For this reason, the update is inversely proportional to the current *energy*

$E_{t,i}$ of the i-th hidden variable, which is $E_{t,i} := \frac{1}{t}\sum_{\tau=1}^{t} y_{\tau,i}^2$. Figure 5.3c shows $\mathbf{w}_1$ after the update for $\mathbf{x}_{t+1}$.

Algorithm 1 TRACKW

Initialise the k hidden variables $\mathbf{w}_i$ to unit vectors $\mathbf{w}_1 = [10\cdots0]^T$, $\mathbf{w}_2 = [010\cdots0]^T$, etc.
Initialise d_i $(i = 1, ...k)$ to a small positive value. Then:
As each point $\mathbf{x}_{t+1}$ arrives, initialise $\acute{\mathbf{x}}_1 := \mathbf{x}_{t+1}$.
for $1 \le i \le k$ **do**
  $y_i := \mathbf{w}_i^T \acute{\mathbf{x}}_i$ $\{(y_{t+1,i} =$ projection onto $\mathbf{w}_i\}$
  $d_i \leftarrow \lambda d_i + y_i^2$ {energy $\propto i$-th eigenval. of $\mathbf{X}_t^T\mathbf{X}_t$}
  $\mathbf{e}_i := \acute{\mathbf{x}}_i - y_i\mathbf{w}_i$ {error, $\mathbf{e}_i \perp \mathbf{w}_i$}
  $\mathbf{w}_i \leftarrow \mathbf{w}_i + \frac{1}{d_i} y_i \mathbf{e}_i$ {update PC estimate}
  $\acute{\mathbf{x}}_{i+1} := \acute{\mathbf{x}}_i - y_i\mathbf{w}_i$ {repeat with remainder of $\mathbf{x}_t$}
end for

The *forgetting factor* λ will be discussed in Section 6.3 (for now, assume $\lambda = 1$). For each i, $d_i = tE_{t,i}$ and $\acute{\mathbf{x}}_i$ is the component of $\mathbf{x}_{t+1}$ in the orthogonal complement of the space spanned by the updated estimates $\mathbf{w}_{i'}, 1 \le i' < i$ of the participation weights. The vectors $\mathbf{w}_i, 1 \le i \le k$ are in order of importance (more precisely, in order of decreasing eigenvalue or energy). It can be shown that, under stationarity assumptions, these $\mathbf{w}_i$ in these equations converge to the true principal directions.

Complexity We only need to keep the k weight vectors $\mathbf{w}_i$ $(1 \le i \le k)$, each n-dimensional. Thus the total cost is $O(nk)$, both in time and of space. The update cost does not depend on t. This is a tremendous gain, compared to the usual PCA computation cost of $O(tn^2)$.

6.2 Detecting the Number of Hidden Variables

In practice, we do not know the number k of hidden variables. We propose to estimate k on the fly, so that we maintain a high percentage f_E of the *energy* E_t. Energy thresholding is a common method to determine how many principal components are needed [33]. Formally, the energy E_t (at time t) of the sequence of $\mathbf{x}_t$ is defined as

$$E_t := \frac{1}{t}\sum_{\tau=1}^{t} \|\mathbf{x}_\tau\|^2 = \frac{1}{t}\sum_{\tau=1}^{t}\sum_{i=1}^{n} x_{\tau,i}^2.$$

Similarly, the energy $\tilde{E}_t$ of the reconstruction $\tilde{\mathbf{x}}$ is defined as

$$\tilde{E}_t := \frac{1}{t}\sum_{\tau=1}^{t} \|\tilde{\mathbf{x}}_\tau\|^2.$$

LEMMA 5.5 *Assuming the* $\mathbf{w}_i, 1 \leq i \leq k$ *are orthonormal, we have*

$$\tilde{E}_t = \frac{1}{t}\sum_{\tau=1}^{t} \|\mathbf{y}_\tau\|^2 = \frac{t-1}{t}\tilde{E}_{t-1} + \frac{1}{t}\|\mathbf{y}_t\|.$$

Proof. If the $\mathbf{w}_i, 1 \leq i \leq k$ are orthonormal, then it follows easily that $\|\tilde{\mathbf{x}}_\tau\|^2 = \|y_{\tau,1}\mathbf{w}_1 + \cdots + y_{\tau,k}\mathbf{w}_k\|^2 = y_{\tau,1}^2\|\mathbf{w}_1\|^2 + \cdots + y_{\tau,k}^2\|\mathbf{w}_k\|^2 = y_{\tau,1}^2 + \cdots + y_{\tau,k}^2 = \|\mathbf{y}_\tau\|^2$ (Pythagorean theorem and normality). The result follows by summing over τ. □

It can be shown that algorithm TRACKW maintains orthonormality without the need for any extra steps (otherwise, a simple re-orthonormalisation step at the end would suffice).

From the user's perspective, we have a low-energy and a high-energy threshold, f_E and F_E, respectively. We keep enough hidden variables k, so the retained energy is within the range $[f_E \cdot E_t,\ F_E \cdot E_t]$. Whenever we get outside these bounds, we increase or decrease k. In more detail, the steps are:

1 Estimate the full energy E_{t+1}, incrementally, from the sum of squares of $x_{\tau,i}$.

2 Estimate the energy $\tilde{E}_{(k)}$ of the k hidden variables.

3 Possibly, adjust k. We introduce a new hidden variable (update $k \leftarrow k+1$) if the current hidden variables maintain too little energy, i.e., $\tilde{E}_{(k)} < f_E E$. We drop a hidden variable (update $k \leftarrow k-1$), if the maintained energy is too high, i.e., $\tilde{E}_{(k)} > F_E E$.

The energy thresholds f_E and F_E are chosen according to recommendations in the literature [33, 20]. We use a lower energy threshold $f_E = 0.95$ and an upper threshold $F_E = 0.98$. Thus, the reconstruction $\tilde{\mathbf{x}}_t$ retains between 95% and 98% of the energy of $\mathbf{x}_t$.

The following lemma proves that the above algorithm guarantees the relative reconstruction error is within the specified interval $[f_E, F_E]$.

LEMMA 5.6 *The relative squared error of the reconstruction satisfies*

$$1 - F_E \leq \frac{\sum_{\tau=1}^{t} \|\tilde{\mathbf{x}}_\tau - \mathbf{x}_\tau\|^2}{\sum_t \|\mathbf{x}_\tau\|^2} \leq 1 - f_E.$$

Proof. From the orthogonality of $\mathbf{x}_\tau$ and $\tilde{\mathbf{x}}_\tau - \mathbf{x}_\tau$ we have $\|\tilde{\mathbf{x}}_\tau - \mathbf{x}_\tau\|^2 = \|\mathbf{x}_\tau\|^2 - \|\tilde{\mathbf{x}}_\tau\|^2 = \|\mathbf{x}_\tau\|^2 - \|\mathbf{y}_\tau\|^2$ (by Lemma 5.5). The result follows by summing over τ and from the definitions of E and $\tilde{E}$. □

Algorithm 2 SPIRIT

Initialize $k \leftarrow 1$
Initialize total energy estimates of $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ per time tick to $E \leftarrow 0$ and $\tilde{E}_1 \leftarrow 0$. Then,
for each new point that arrives **do**
  Update $\mathbf{w}_i$, for $1 \leq i \leq k$ (TRACKW).
  Update the estimates (for $1 \leq i \leq k$)

$$E \leftarrow \frac{(t-1)E + \|\mathbf{x}_t\|^2}{t} \quad \text{and} \quad \tilde{E}_i \leftarrow \frac{(t-1)\tilde{E}_i + y_{t,i}^2}{t}.$$

  Let the estimate of retained energy be

$$\tilde{E}_{(k)} := \sum_{i=1}^{k} \tilde{E}_i.$$

  if $\tilde{E}_{(k)} < f_E E$ **then**
    Start estimating $\mathbf{w}_{k+1}$ (initialising as in TRACKW)
    Initialise $\tilde{E}_{k+1} \leftarrow 0$
    Increase $k \leftarrow k+1$.
  end if
  if $\tilde{E}_{(k)} > F_E E$ **then**
    Discard $\mathbf{w}_k$ and $\tilde{E}_k$
    Decrease $k \leftarrow k-1$
  end if
end for

6.3 Exponential Forgetting

We can adapt to more recent behavior by using an exponential forgetting factor, $0 < \lambda < 1$. This allows us to follow trend drifts over time. We use the same λ for the estimation of both $\mathbf{w}_i$ and of the AR models (see Section 7.1). However, we also have to properly keep track of the energy, discounting it with the same rate, i.e., the update at each step is:

$$E \leftarrow \frac{\lambda(t-1)E + \|\mathbf{x}_t\|^2}{t} \quad \text{and} \quad \tilde{E}_i \leftarrow \frac{\lambda(t-1)\tilde{E}_i + y_{t,i}^2}{t}.$$

Typical choices are $0.96 \leq \lambda \leq 0.98$ [27]. As long as the values of $\mathbf{x}_t$ do not vary wildly, the exact value of λ is not crucial. We use $\lambda = 0.96$ throughout. A value of $\lambda = 1$ makes sense when we know that the sequence is stationary (rarely true in practice, as most sequences gradually drift). Note that the value of λ does not affect the computation cost of our method. In this sense, an exponential forgetting factor is

more appealing than a sliding window, as the latter has explicit buffering requirements.

7. An Application-driven View: Putting Correlations to Work

We show how we can exploit the correlations and hidden variables to do (a) forecasting, (b) missing value estimation, (c) summarization of the large number of streams into a small, manageable number of hidden variables, and (d) outlier detection. We will use SPIRIT because of its hidden variable-based approach which makes it very convenient for such tasks, though many of these tasks can also be accomplished by other methods.

7.1 Forecasting and Missing Values

The hidden variables $\mathbf{y}_t$ give us a much more compact representation of the "raw" variables $\mathbf{x}_t$, with guarantees of high reconstruction accuracy (in terms of relative squared error, which is less than $1 - f_E$). When our streams exhibit correlations, as we often expect to be the case, the number k of the hidden variables is much smaller than the number n of streams. Therefore, we can apply *any* forecasting algorithm to the vector of hidden variables $\mathbf{y}_t$, instead of the raw data vector $\mathbf{x}_t$. This reduces the time and space complexity by orders of magnitude, because typical forecasting methods are quadratic or worse on the number of variables.

In particular, we fit the forecasting model on the $\mathbf{y}_t$ instead of $\mathbf{x}_t$. The model provides an estimate $\hat{\mathbf{y}}_{t+1} = f(\mathbf{y}_t)$ and we can use this to get an estimate for

$$\hat{\mathbf{x}}_{t+1} := \hat{y}_{t+1,1}\mathbf{w}_1[t] + \cdots + \hat{y}_{t+1,1}\mathbf{w}_k[t],$$

using the weight estimates $\mathbf{w}_i[t]$ from the previous time tick t. We chose auto-regression for its intuitiveness and simplicity, but any online method can be used.

Correlations Since the principal directions are orthogonal to one another ($\mathbf{w}_i \perp \mathbf{w}_j, i \neq j$), the components of $\mathbf{y}_t$ are *by construction uncorrelated*—the correlations have already been captured by the $\mathbf{w}_i, 1 \leq i \leq k$. We can take advantage of this de-correlation reduce forecasting complexity. In particular for auto-regression, we found that one AR model per hidden variable provides results comparable to multivariate AR.

Auto-regression Space complexity for multivariate AR (e.g., MUSCLES [58]) is $O(n^3\ell^2)$, where ℓ is the auto-regression window length.

For AR per stream (ignoring correlations), it is $O(n\ell^2)$. However, for SPIRIT, we need $O(kn)$ space for the $\mathbf{w}_i$ and, with one AR model per y_i, the total space complexity is $O(kn + k\ell^2)$. As published, MUSCLES requires space that grows cubically with respect to the number of streams n. We believe it can be made to work with quadratic space, but this is still prohibitive. Both AR per stream and SPIRIT require space that grows linearly with respect to n, but in SPIRIT k is typically very small ($k \ll n$) and, in practice, SPIRIT requires less memory and time per update than AR per stream. More importantly, a single, independent AR model per stream cannot capture *any* correlations, whereas SPIRIT indirectly exploits the correlations present *within* a time tick.

Missing values When we have a forecasting model, we can use the forecast based on $\mathbf{x}_{t-1}$ to estimate missing values in $\mathbf{x}_t$. We then use these estimated missing values to update the weight estimates, as well as the forecasting models. Forecast-based estimation of missing values is the most time-efficient choice and gives very good results.

7.2 Interpretation

At *any* given time t, SPIRIT readily provides two key pieces of information (aside from the forecasts, etc.): (i)The number of hidden variables k. (ii) The weights $w_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq n$. Intuitively, the magnitude $|w_{i,j}|$ of each weight tells us how much the i-th hidden variable contributes to the reconstruction of the j-th stream.

In the chlorine example during phase 1 (see Figure 5.1), the dataset has only one hidden variable, because one sinusoidal-like pattern can reconstruct both streams (albeit with different weights for each). Thus, SPIRIT correctly identifies correlated streams. When the correlation was broken, SPIRIT introduces enough hidden variables to capture that. Finally, it also spots that, in phase 3, normal operation is reestablished and thus disposes of the unnecessary hidden variable.

8. Pattern Discovery across Time

Many pattern discovery methods first project the data onto "all" bases in a given family (e.g., Fourier, wavelets, etc) and then choose a few coefficients that capture the most information. In contrast, among *all* possible bases, we first choose a few bases that are guaranteed to capture the most information and consequently project the data only onto those. However, efficiently determining these few bases and incrementally updating them as new points arrive is a challenging problem. To

that end, we use techniques related to those for discovering correlations across streams, illustrating the relationship between the two problems.

Let us first assume that someone gives us a window size. Then, the problem we want to solve is the following:

PROBLEM 8.1 (FIXED-WINDOW OPTIMAL PATTERNS) *Given a time series x_t, $t = 1, 2, \ldots$ and a window size w, find the patterns that best summarize the series at this window size.*

The patterns are w-dimensional vectors $\mathbf{v}_i \equiv [v_{i,1}, \ldots, v_{i,w}]^T \in \mathbb{R}^w$, chosen so that they capture "most" of the information in the series (in a way that we will make precise later).

In practice, however, we do not know *a priori* the right window size. Therefore, with respect to the second requirement (multi-scale), we want to solve the following problem:

PROBLEM 8.2 (OPTIMAL LOCAL PATTERNS) *Given a time series x_t and a set of windows $\mathcal{W} := \{w_1, w_2, w_3, \ldots\}$, find (i) the optimal patterns for each of these, and (ii) the best window w^* to describe the key patterns in the series.*

So how do we go about finding these patterns? An elementary concept we need to introduce is *time-delay coordinates*. We are given a time series x_t, $t = 1, 2, \ldots$ with m points seen so far. Intuitively, when looking for patterns of length w, we divide the series in consecutive, non-overlapping subsequences of length w. Thus, if the original series is a $m \times 1$ matrix (not necessarily materialized), we substitute it with a $\frac{m}{w} \times w$ matrix. Instead of m scalar values we now have a sequence of m/w vectors with dimension w. It is natural to look for patterns among these time-delay vectors.

DEFINITION 5.7 (DELAY COORDINATES) *Given a sequence denoted by $\mathbf{x} \equiv [x_1, x_2, \ldots, x_t, \ldots, x_m]^T$ and a* delay *(or window) w, the* delay coordinates *are a $\lceil m/w \rceil \times w$ matrix with the t'-th row equal to $\mathbf{X}^{(w)}_{(t')} := [x_{(t'-1)w+1}, x_{(t'-1)w+2}, \ldots, x_{t'w}]^T$.*

Of course, neither $\mathbf{x}$ nor $\mathbf{X}^{(w)}$ need to be fully materialized at any point in time. In practice, we only need to store the last row of $\mathbf{X}^{(w)}$.

Also, note that we choose non-overlapping windows. We could also use overlapping windows, in which case $\mathbf{X}^{(w)}$ would have $m - w + 1$ rows, with row t consisting of values $x_t, x_{t+1}, \ldots, x_{t+w}$. In this case, there are some subtle differences [23], akin to the differences between "standard" wavelets and *maximum-overlap* or *redundant* wavelets [46]. However, in practice non-overlapping windows are equally effective for pattern

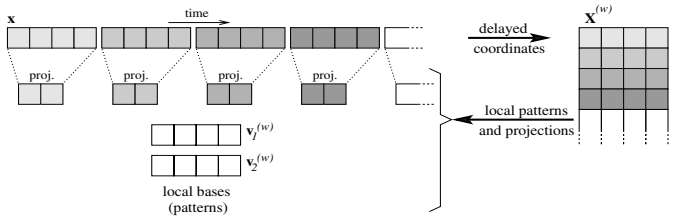


Figure 5.4. Illustration of local patterns for a fixed window (here, $w = 4$).

discovery and also lend themselves better to incremental, streaming estimation using limited resources.

More generally, the original time series does not have to be scalar, but can also be vector-valued itself. We still do the same, only each row of $\mathbf{X}^{(w)}$ is now a concatenation of rows of $\mathbf{X}$ (instead of a concatenation of scalar values). More precisely, we construct the general time-delay coordinate matrix as follows:

Procedure 1 DELAY ($\mathbf{X} \in \mathbb{R}^{m \times n}$, w)

$m' \leftarrow \lfloor m/w \rfloor$ and $n' \leftarrow nw$
Output is $\mathbf{X}^{(w)} \in \mathbb{R}^{m' \times n'}$ {not necessarily materialized}
for $t = 1$ to m' **do**
  Row $\mathbf{X}^{(w)}_{(t)} \leftarrow$ concatenation of rows
    $\mathbf{X}_{((t-1)w+1)}, \mathbf{X}_{((t-1)w+2)}, \cdots \mathbf{X}_{(tw)}$
end for

Incremental SVD The SVD update algorithm used in SPIRIT can be applied incrementally to vectors that represent windows of the same time series. As we have seen, its accuracy is good, while it does not need to store the left singular vectors. Since our goal is to find patterns at multiple scales without an upper bound on the window size, this is a more suitable choice. Furthermore, if we need to place more emphasis on recent trends, it is rather straightforward to incorporate an exponential forgetting scheme, which works well in practice [44]. For each new row, the algorithm updates $k \cdot n$ numbers, so total space requirements are $O(nk)$ and the time per update is also $O(nk)$. Finally, the incremental update algorithms need only the observed values and can therefore easily handle missing values by imputing them based on current estimates of the singular vectors.

8.1 Locally Optimal Patterns

We now have all the pieces in place to answer the first question: for a given window w, how do we find the locally optimal patterns? Figure 5.4 illustrates the main idea. Starting with the original time series $\mathbf{x}$, we transfer to time-delay coordinates $\mathbf{X}^{(w)}$. The local patterns are the right singular vectors of $\mathbf{X}^{(w)}$, which are optimal in the sense that they minimize the total squared approximation error of the rows $\mathbf{X}^{(w)}_{(i)}$. The detailed algorithm is shown below.

Algorithm 3 LocalPattern ($\mathbf{x} \in \mathbb{R}^m$, w, $k = 3$)

Use delay coord. $\mathbf{X}^{(w)} \leftarrow \text{Delay}(\mathbf{x}, w)$
Compute SVD of $\mathbf{X}^{(w)} = \mathbf{U}^{(w)}\mathbf{\Sigma}^{(w)}\mathbf{V}^{(w)}$
Local patterns are $\mathbf{v}_1^{(w)}, \ldots, \mathbf{v}_k^{(w)}$
Power is $\pi^{(w)} \leftarrow \sum_{i=k+1}^{w} \sigma_i^2 / w = \left(\sum_{t=1}^{m} x_t^2 - \sum_{i=1}^{k} \sigma_i^2\right)/w$
$\tilde{\mathbf{P}}^{(w)} \leftarrow \tilde{\mathbf{U}}^{(w)}\tilde{\mathbf{\Sigma}}^{(w)}$ {low-dim. proj. onto local patterns}
return $\tilde{\mathbf{V}}^{(w)}$, $\tilde{\mathbf{P}}^{(w)}$, $\tilde{\mathbf{\Sigma}}^{(w)}$ and $\pi^{(w)}$

For now, the projections $\tilde{\mathbf{P}}^{(w)}$ onto the local patterns $\tilde{\mathbf{v}}_{(i)}$ are not needed, but we will use them later. Also, note that LocalPattern can be applied in general to n-dimensional vector-valued series. The pseudocode is the same, since Delay can also operate on matrices $\mathbf{X} \in \mathbb{R}^{m \times n}$. The reason for this will also become clear, but for now it suffices to observe that the first argument of LocalPattern may be a matrix, with one row $\mathbf{x}_{(t)} \in \mathbb{R}^n$ per timestamp $t = 1, 2, \ldots, m$.

When computing the SVD, we really need only the highest k singular values and the corresponding singular vectors, because we only return $\tilde{\mathbf{V}}^{(w)}$ and $\tilde{\mathbf{P}}^{(w)}$. Therefore, we can avoid computing the full SVD and use somewhat more efficient algorithms, just for the quantities we actually need.

Also, note that $\tilde{\mathbf{\Sigma}}^{(w)}$ can be computed from $\tilde{\mathbf{P}}^{(w)}$, since by construction

$$\sigma_i^2 = \|\mathbf{p}_i\|^2 = \sum_{j=1}^{m} p_{ji}^2. \tag{5.5}$$

However, we return these separately, which avoids duplicate computation. More importantly, when we later present our streaming approach, we won't be materializing $\tilde{\mathbf{P}}^{(w)}$. Furthermore, Equation (5.5) does not hold exactly for the estimates returned by IncrementalSVD and it is better to use the estimates of the singular values σ_i^2 computed as part of IncrementalSVD.

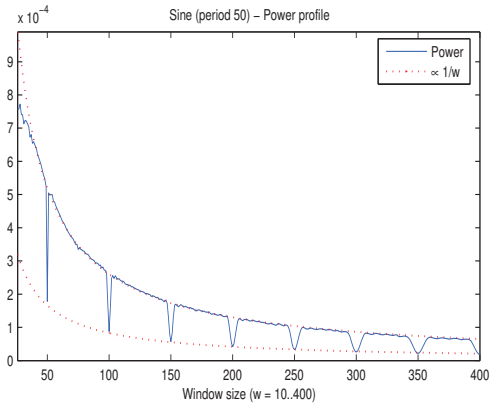


Figure 5.5. Power profile of sine wave $x_t = \sin(2\pi t/50) + \epsilon_t$, with Gaussian noise $\epsilon_t \sim \mathcal{N}(5, 0.5)$.

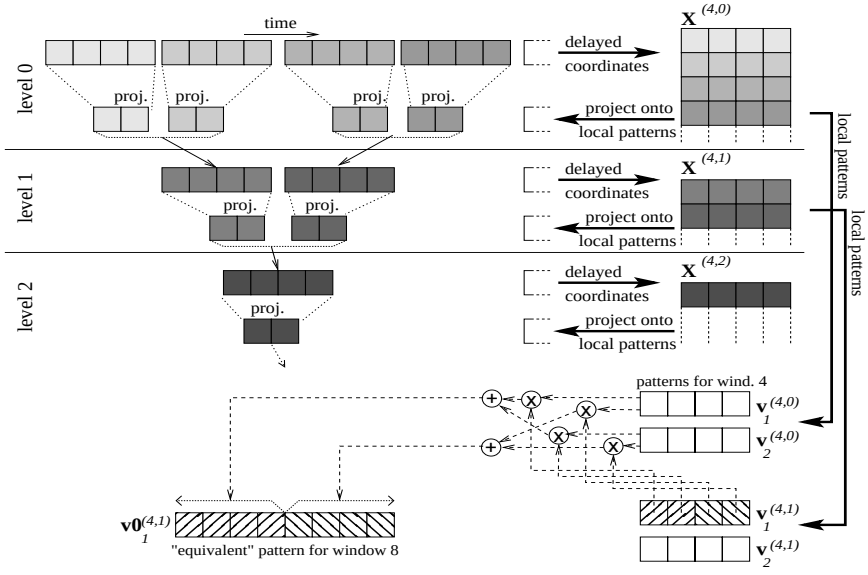


Figure 5.6. Multi-scale pattern discovery (hierarchical, $w_0 = 4$, $W = 2$, $k = 2$).

8.1.1 Power Profile. Next, let us assume we have optimal local patterns for a number of different window sizes. Which of these windows is the best to describe the main trends? Intuitively, the key idea is that if there is a trend that repeats with a period of T, then different subsequences in the time-delay coordinate space should be highly correlated when $w \approx T$. Although the trends can be arbitrary, we illustrate the intuition with a sine wave, in Figure 5.5. The plot shows the squared approximation error per window element, using $k = 1$ pat-

tern on a sine wave with period $T = 50$. As expected, for window size $w = T = 50$ the approximation error drops sharply and essentially corresponds to the Gaussian noise floor. Naturally, for windows $w = iT$ that are multiples of T the error also drops. Finally, observe that the error for all windows is proportional to $\frac{1}{w}$, since it is *per window element*. Eventually, for window size equal to the length of the entire time series $w = m$ (not shown in Figure 5.5, where $m = 2000$), we get $\pi^{(m)} = 0$ since first pattern is the only singular vector, which coincides with the series itself, so the residual error is zero.

Formally, the squared approximation error of the time-delay matrix $\mathbf{X}^{(w)}$ is

$$\epsilon^{(w)} := \sum_t \|\tilde{\mathbf{x}}_{(t)}^{(w)} - \mathbf{x}_{(t)}^{(w)}\|^2 = \|\tilde{\mathbf{X}}^{(w)} - \mathbf{X}^{(w)}\|_F^2,$$

where $\tilde{\mathbf{X}}^{(w)} := \tilde{\mathbf{P}}^{(w)}(\tilde{\mathbf{V}}^{(w)})^T$ is the reconstruction and $\|\mathbf{A}\|_F^2 := \sum_{i,j} a_{ij}^2$ denotes the Frobenius norm of $\mathbf{A}$ (sum of squares of matrix entries a_{ij}). From the definition of the SVD-based approximation error (see Section 3), as well as the fact that the sum of squares of the singular values of a matrix is equal to the sum of squares of its values, we have

$$\epsilon^{(w)} = \|\mathbf{X}^{(w)}\|_F^2 - \|\tilde{\mathbf{P}}^{(w)}\|_F^2 \approx \|\mathbf{x}\|^2 - \sum_{i=1}^{k} \left(\sigma_i^{(w)}\right)^2.$$

Based on this, we define the power, which is an estimate of the error per window element.

Definition 5.8 (Power profile $\pi^{(w)}$) *For a given number of patterns ($k = 2$ or 3) and for any window size w, the power profile is the sequence defined by*

$$\pi^{(w)} := \frac{\epsilon^{(w)}}{w}. \tag{5.6}$$

More precisely, this is an estimate of the variance per dimension, assuming that the discarded dimensions correspond to isotropic Gaussian noise (i.e., uncorrelated with same variance in each dimension) [52]. As explained, this will be much lower when $w = T$, where T is the period of an arbitrary main trend.

The following lemma follows from the above observations. Note that the conclusion is valid both ways, i.e., perfect copies imply zero power and vice versa. Also, the conclusion holds regardless of alignment (i.e., the periodic part does not have to start at the begining of a windowed subsequence). A change in alignment will only affect the phase of the discovered local patterns, but not their shape or the reconstruction accuracy.

Observation 8.1 (Zero power) *If $\mathbf{x} \in \mathbb{R}^t$ consists of exact copies of a subsequence of length T then, for every number of patterns $k = 1, 2, \ldots$ and at each multiple of T, we have $\pi^{(iT)} = 0$, $i = 1, 2, \ldots$, and vice versa.*

In general, if the trend does not consist of exact copies, the power will not be zero, but it will still exhibit a sharp drop. We exploit precisely this fact to choose the "right" window.

Choosing the window Next, we state the steps for interpreting the power profile to choose the appropriate window that best captures the main trends: (i) Compute the power profile $\pi^{(w)}$ versus w; (ii) Look for the first window w_0^* that exhibits a sharp drop in $\pi^{(w_0^*)}$ and ignore all other drops occurring at windows $w \approx iw_0^*$, $i = 2, 3, \ldots$ that are approximately multiples of w_0^*; (iii) If there are several sharp drops at windows w_i^* that are *not* multiples of each other, then any of these is suitable. We simply choose the smallest one; alternatively, we could choose based on prior knowledge about the domain if available, but that is not necessary; (iv) If there are no sharp drops, then no strong periodic/cyclic components are present. However, the local patterns at any window can still be examined to gain a picture of the time series behavior.

8.2 Multiple-Scale Patterns

In this section we tackle the following question: how do we efficiently compute the optimal local patterns for multiple windows (as well as the associated power profiles), so as to quickly zero in to the "best" window size? First, we can choose a geometric progression of window sizes: rather than estimating the patterns for windows of length w_0, $w_0 + 1$, $w_0 + 2$, $w_0 + 3, \ldots$, we estimate them for windows of w_0, $2w_0$, $4w_0, \ldots$ or, more generally, for windows of length $w_l := w_0 \cdot W^l$ for $l = 0, 1, 2, \ldots$. Thus, the size of the window set $\mathcal{W}$ we need to examine is dramatically reduced. Still, this is (i) computationally expensive (for each window we still need $O(ktw)$ time and, even worse, (ii) still requires buffering all the points (needed for large window sizes, close to the time series length). Next, we show how we can reduce complexity even further.

8.2.1 Hierarchical SVD. The main idea of our approach to solve this problem is shown in Figure 5.6. Let us assume that we have, say $k = 2$ local patterns for a window size of $w_0 = 100$ and we want to compute the patterns for window $w^{(100,1)} = 100 \cdot 2^1 = 200$. The naive approach is to construct $\mathbf{X}^{(200)}$ from scratch and compute the SVD. However, we can reuse the patterns found from $\mathbf{X}^{(100)}$. Using the $k = 2$ patterns $\mathbf{v}_1^{(100)}$ and $\mathbf{v}_2^{(100)}$ we can reduce the first $w_0 = 100$ points $x_1, x_2, \ldots, x_{100}$ into just two points, namely their projections $p_{1,1}^{(100)}$ and $p_{1,2}^{(100)}$ onto $\mathbf{v}_1^{(100)}$ and $\mathbf{v}_2^{(100)}$, respectively. Similarly, we can reduce

the next $w_0 = 100$ points $x_{101}, x_{102}, \ldots, x_{200}$ also into two numbers, $p_{2,1}^{(100)}$ and $p_{2,2}^{(100)}$, and so on. These projections, by construction, approximate the original series well. Therefore, we can represent the first row $\mathbf{x}_{(1)}^{(200)} \equiv [x_1, \ldots, x_{200}]^T \in \mathbb{R}^{200}$ of $\mathbf{X}^{(200)}$ with just four numbers, $\mathbf{x}_{(1)}^{(100,1)} \equiv \left[p_{1,1}^{(100)}, p_{1,2}^{(100)}, p_{2,1}^{(100)}, p_{2,2}^{(100)}\right]^T \in \mathbb{R}^4$. Doing the same for the other rows of $\mathbf{X}^{(200)}$, we construct a matrix $\mathbf{X}^{(100,1)}$ with just $n = 4$ columns, which is a very good approximation of $\mathbf{X}^{(200)}$.Consequently, we compute the local patterns using $\mathbf{X}^{(100,1)}$ instead of $\mathbf{X}^{(200)}$. Repeating this process recursively, we can find the local patterns for a window $w^{(100,2)} = 100 \cdot 2^2 = 400$ and so on.

DEFINITION 5.9 (LEVEL-(w_0, l) WINDOW) *The level-(w_0, l) window corresponds to an original window size (or* scale*) $w_l := w_0 \cdot W^l$. Patterns at each level l are found recursively, using patterns from the previous level $l - 1$.*

In the above example, we have $w_0 = 100$ and $l = 0, 1$. Since w_0 and W are fixed for a particular sequence of scales w_l, we will simply refer to level-l windows and patterns. The recursive construction is based on the level-l delay matrix and corresponding patterns.

DEFINITION 5.10 (LEVEL-l DELAY MATRIX $\mathbf{X}^{(w_0,l)}$) *Given a starting window w_0 and a scale factor W, the level-l delay matrix is simply $\mathbf{X}^{(w_0,0)} := \mathbf{X}^{(w_0)}$ for $l = 0$ and for $l = 1, 2, \ldots$ it is recursively defined by*

$$\mathbf{X}^{(w_0,l)} := \textsc{Delay}\big(\tilde{\mathbf{P}}^{(w_0,l-1)}, W\big),$$

where $\tilde{\mathbf{P}}^{(w_0,l)} := \mathbf{X}^{(w_0,l)}\tilde{\mathbf{V}}^{(w_0,l)}$ is the projection onto the level-l patterns $\tilde{\mathbf{V}}^{(w_0,l)}$ which are found based on $\mathbf{X}^{(w_0,l)}$. The level-l delay matrix is an approximation of the delay matrix $\mathbf{X}^{(w_l)}$ for window size $w_l = w_0 W^l$.

In our example, the patterns extracted from $\mathbf{X}^{(100,1)}$ are four-dimensional vectors, $\mathbf{v}_i^{(100,1)} \in \mathbb{R}^4$, whereas the patterns for $\mathbf{X}^{(200)}$ would be 200-dimensional vectors $\mathbf{v}_i^{(200)} \in \mathbb{R}^{200}$. However, we can appropriately combine $\mathbf{v}_i^{(100,1)}$ and $\mathbf{v}_i^{(100,0)} \equiv \mathbf{v}_i^{(100)}$ to estimate $\mathbf{v}_i^{(200)}$.

DEFINITION 5.11 (LEVEL-l LOCAL PATTERN $\mathbf{v0}_i^{(w_0,l)}$) *The level-l pattern $\mathbf{v0}_i^{(w_0,l)}$, for all $i = 1, 2, \ldots, k$, corresponding to a window of $w_l = w_0 W^l$ is simply $\mathbf{v0}_i^{(w_0,0)} := \mathbf{v}_i^{(w_0)}$ for $l = 0$ and for $l = 1, 2, \ldots$ it is defined recursively by*

$$\mathbf{v0}_i^{(w_0,l)}[(j-1)w_{l-1} + 1 : jw_{l-1}] := \mathbf{V0}^{(w_0,l-1)}\big(\mathbf{v}_i^{(w_0,l)}[(j-1)k + 1 : jk]\big), \quad (5.7)$$

for $j = 1, 2, \ldots, W$. It is an approximation of the local patterns $\mathbf{v}_i^{(w_l)}$ of the original delay matrix $\mathbf{X}^{(w_l)}$, for window size $w_l = w_0 W^l$.

Consider $\mathbf{v0}_1^{(100,1)}$ in our example. The first $k = 2$ out of $kW = 4$ numbers in $\mathbf{v}_1^{(100,1)}$ approximate the patterns among the 2-dimensional vectors $\mathbf{p}_{(j)}^{(100,0)}$, which in turn capture patterns among the 100-dimensional vectors $\mathbf{x}_{(i)}^{(100,0)}$ of the original time-delay matrix. Thus, but forming the appropriate linear combination of the 100-dimensional patterns $\mathbf{v}_i^{(100,0)} \equiv \mathbf{v0}_i^{(100,0)}$ (i.e., the columns of $\tilde{\mathbf{V}}^{(100,0)} \equiv \mathbf{V0}^{(100,0)}$), weighted according to $\mathbf{v}_1^{(100,1)}[1:2]$, we can construct the first half of the 200-dimensional pattern $\mathbf{v0}_1^{(100,1)}[1:100]$ (left-slanted entries in Figure 5.6). Similarly, a linear combination of the columns of $\tilde{\mathbf{V}}^{(100,0)} \equiv \mathbf{V0}^{(100,0)}$ weighted according to $\mathbf{v}_1^{(100,1)}[3:4]$ gives us the second half of the 200-dimensional pattern $\mathbf{v0}_1^{(100,1)}[101:200]$ (right-slanted entries in Figure 5.6). For level $l = 2$ we similarly combine the columns of $\mathbf{V0}^{(100,1)}$ according to $\mathbf{v}_1^{(100,2)}[1:2]$ (for the first half, $\mathbf{v0}_1^{(100,2)}[1:200]$) and to $\mathbf{v}_[^{(100,2)}3:4]$ (for the second half, $\mathbf{v0}_1^{(100,2)}[201:400]$) and so on, for the higher levels.

LEMMA 5.12 (ORTHONORMALITY OF $\mathbf{v0}_i^{(w_0,l)}$) *We have* $\|\mathbf{v0}_i^{(w_0,l)}\| = 1$ *and, for* $i \neq j$, $\left(\mathbf{v0}_i^{(w_0,l)}\right)^T \left(\mathbf{v0}_j^{(w_0,l)}\right) = 0$, *where* $i, j = 1, 2, \ldots, k$.

Proof. For level $l = 0$ they are orthonormal since they coincide with the original patterns $\mathbf{v}_i^{(w_0)}$ which are by construction orthonormal. We proceed by induction on the level $l \geq 1$. Without loss of generality, assume that $k = 2$ and, for brevity, let $\mathbf{B} \equiv \mathbf{V0}^{(w_0,l-1)}$ and $\mathbf{b}_{i,1} \equiv \mathbf{v}_i^{(w_0,l)}[1:k]$, $\mathbf{b}_{i,2} \equiv \mathbf{v}_i^{(w_0,l)}[k+1:k]$, so that $\mathbf{v}_i^{(w_0,l)} = [\mathbf{b}_{i,1}, \mathbf{b}_{i,2}]$. Then

$$\begin{aligned}\|\mathbf{v0}_i^{(w_0,l)}\|^2 &= [\mathbf{B}\mathbf{b}_{i,1}\ \ \mathbf{B}\mathbf{b}_{i,2}]^2 = \|\mathbf{B}\mathbf{b}_{i,1}\|^2 + \|\mathbf{B}\mathbf{b}_{i,2}\|^2 \\ &= \|\mathbf{b}_{i,1}\|^2 + \|\mathbf{b}_{i,2}\|^2 = \|\mathbf{v}_i^{(w_0,l)}\|^2 = 1,\end{aligned}$$

and

$$\begin{aligned}\left(\mathbf{v0}_i^{(w_0,l)}\right)^T \left(\mathbf{v0}_j^{(w_0,l)}\right) &= [\mathbf{B}\mathbf{b}_{i,1}\ \ \mathbf{B}\mathbf{b}_{i,2}]^T [\mathbf{B}\mathbf{b}_{j,1}\ \ \mathbf{B}\mathbf{b}_{j,2}] \\ &= \mathbf{b}_{i,1}^T \mathbf{B}^T \mathbf{B}\mathbf{b}_{j,1} + \mathbf{b}_{i,2}^T \mathbf{B}^T \mathbf{B}\mathbf{b}_{j,2} \\ &= \mathbf{b}_{i,1}^T \mathbf{b}_{j,1} + \mathbf{b}_{i,2}^T \mathbf{b}_{j,2} \\ &= \left(\mathbf{v}_i^{(w_0,l)}\right)^T \left(\mathbf{v}_j^{(w_0,l)}\right) = 0,\end{aligned}$$

since $\mathbf{B}$ preserves dot products as an orthonormal matrix (by inductive hypothesis) and $\mathbf{v}_i^{(w_0,l)}$ are orthonormal by construction. □

The detailed hierarchical SVD algorithm is shown below. In practice, the maximum level L is determined based on the length m of the time series so far, $L \approx \log_W(m/w_0)$.

Algorithm 4 HIERARCHICAL ($\mathbf{x} \in \mathbb{R}^m$, w_0, W, L, $k = 6$)

{Start with level $l = 0$, corresponding to window w_0}
$\tilde{\mathbf{V}}^{(w_0,0)}$, $\tilde{\mathbf{P}}^{(w_0,0)}$, $\tilde{\mathbf{\Sigma}}^{(w_0,0)}$, $\pi^{(w_0,0)} \leftarrow$
LOCALPATTERN($\mathbf{x}, w_0, k$)
{Levels l, corresponding to window $w_l = w_0 \cdot W^l$}
for level $l = 1$ to L **do**
  $\tilde{\mathbf{V}}^{(w_0,l)}$, $\tilde{\mathbf{P}}^{(w_0,l)}$, $\tilde{\mathbf{\Sigma}}^{(w_0,l)}$, $\pi^{(w_0,l)} \leftarrow$
  LOCALPATTERN($\tilde{\mathbf{P}}^{(w_0,l-1)}, W, k$)
  Compute patterns $\mathbf{v0}_i^{(w_0,l)}$ for window size w_l are based on Equation (5.7)
end for

Choosing the initial window The initial window w_0 has some impact on the quality of the approximations. This also depends on the relationship of k to w_0 (the larger k is, the better the approximation and if $k = w_0$ then $\tilde{\mathbf{P}}^{(w_0,1)} = \mathbf{X}^{(w_0)}$, i.e., no information is discarded at the first level). However, we want k to be relatively small since, as we will see, it determines the buffering requirements of the streaming approach. Hence, we fix $k = 6$. We found that this simple choice works well for real-world sequences, but we could also use energy-based thresholding [33], which can be done incrementally.

If w_0 is too small, then we discard too much of the variance too early. If w_0 is unnecessarily big, this increases buffering requirements and the benefits of the hierarchical approach diminish. In practice, a good compromise is a value in the range $10 \leq w_0 \leq 20$.

Finally, out of the six patterns we keep per level, the first two or three are of interest and reported to the user. The remaining are kept to ensure that $\mathbf{X}^{(w_0,l)}$ is a good approximation of $\mathbf{X}^{(w_l)}$.

Choosing the scales As discussed in Section 8.1.1, if there is a sharp drop of $\pi^{(T)}$ at window $w = T$, then we will also observe drops at multiples $w = iT$, $i = 2, 3, \ldots$. Therefore, we choose a few different starting windows w_0 and scale factors W that are relatively prime to each other. In practice, the following set of three choices is sufficient

to quickly zero in on the best windows and the associated optimal local patterns:

$$k = 6 \quad \text{and} \quad (w_0, W) \in \{(9,2),(10,2),(15,3)\}$$

Complexity For a total of $L \approx \log_W(t/w_0) = O(\log t)$ levels we have to compute the first k singular values and vectors of $\mathbf{X}^{(w_0,l)} \in \mathbb{R}^{t/(w_0 W^l) \times Wk}$, for $l = 1, 2, \ldots$. A batch SVD algorithm requires time $O\big(k \cdot (Wk)^2 \cdot \frac{t}{w_0 W^l}\big)$, which is $O\big(\frac{W^2 k^2 t}{W^l}\big)$ since $k < w_0$. Summing over $l = 1, \ldots, L$, we get $O(W^2 k^2 t)$. Finally, for $l = 0$, we need $O\big(k \cdot w_0^2 \frac{t}{w_0}\big) = O(k w_0 t)$. Thus, the total complexity is $O(W^2 k^2 t + k w_0 t)$. Since W and w_0 are fixed, we finally have the following

LEMMA 5.13 (BATCH HIERARCHICAL COMPLEXITY) *The total time for the hierarchical approach is $O(k^2 t)$, i.e., linear with respect to the time series length.*

This is a big improvement over the $O(t^3 k)$ time of the non-hierarchical approach. However, we still need to buffer all the points. We address this problem in the next section.

8.3 Streaming Computation

In this section we explain how to perform the necessary computations in an incremental, streaming fashion. We designed our models precisely to allow this step. The main idea is that we recursively invoke only one iteration of each loop in INCREMENTALSVD (for LOCALPATTERN) and in HIERARCHICAL, as soon as the necessary number of points has arrived. Subsequently, we can discard these points and proceed with the next non-overlapping window.

Modifying LocalPattern We buffer consecutive points of $\mathbf{x}$ (or, in general, rows of $\mathbf{X}$) until we accumulate w of them, forming one row of $\mathbf{X}^{(w)}$. At that point, we can perform one iteration of the outer loop in INCREMENTALSVD to update all k local patterns. Then, we can discard the w points (or rows) and proceed with the next w. Also, since on higher levels the number of points for SVD may be small and close to k, we may choose to initially buffer just the first k rows of $\mathbf{X}^{(w)}$ and use them to bootstrap the SVD estimates, which we subsequently update as described.

Modifying Hierarchical For level $l = 0$ we use the modified LOCALPATTERN on the original series, as above. However, we also store the k projections onto the level-0 patterns. We buffer W consecutive sets of

these projections and as soon as kW values accumulate, we update the k local patterns for level $l = 1$. Then we can discard the kW projections from level-0, but we keep the k level-1 projections. We proceed in the same way for all other levels $l \geq 2$.

Complexity Compared to the batch computation, we need $O\big(k \cdot Wk \cdot \frac{t}{w_0 W^l}\big) = O\big(\frac{kt}{W^{l-1}}\big)$ time to compute the first k singular values and vectors of $\mathbf{X}^{(w_0,l)}$ for $l = 1, 2, \ldots$. For $l = 0$ we need $O\big(k \cdot w_0 \cdot \frac{t}{w_0}\big) = O(kt)$ time. Summing over $l = 0, 1, \ldots, L$ we get $O(kt)$. With respect to space, we need to buffer w_0 points for $l = 0$ and Wk points for each of the remaining $L = O(\log t)$ levels, for a total of $O(k \log t)$. Therefore, we have the following

LEMMA 5.14 (STREAMING, HIER. COMPLEXITY) *Amortized cost is* $O(k)$ *per incoming point and total space is* $O(k \log t)$.

Since $k = 6$, the update time is constant per incoming point and the space requirements grow logarithmically with respect to the size t of the series. Table 5.2 summarizes the time and space complexity for each approach.

	Time		Space	
	Non-hier.	Hier.	Non-hier.	Hier
Batch	$O(t^3k)$	$O(tk^2)$	all	all
Incremental	$O(t^2k)$	$O(tk)$	$O(t)$	$O(k \log t)$

Table 5.2. Summary of time and space complexity.

9. Conclusions

This chapter surveyed techniques for dimensionality pattern discovery across multiple streams (correlation detection and streaming dimensionality reduction) as well as across time within a single stream (auto-correlation detection and filtering/compression), presenting a unified view of these two central problems. The chapter overviewed fundamental techniques, including auto-regression, principal component analysis, and the singular value decomposition, and shown what it takes to apply these ideas consistently yet effectively to tackle both types of problems on time series stream and sensor data. Furthermore, a discussion of broadly related work in the area of time series stream mining was presented.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, 2003.

[2] C. C. Aggarwal, Y. Xie, and P. S. Yu. On dynamic data-driven selection of sensor streams. In *KDD*, 2011.

[3] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, 1993.

[4] M. H. Ali, M. F. Mokbel, W. Aref, and I. Kamel. Detection and tracking of discrete phenomena in sensor network databases. In *SSDBM*, 2005.

[5] I. Assent, P. Kranen, C. Baldauf, and T. Seidl. Detecting outliers on arbitrary data streams using anytime approaches. In *StreamKDD*, 2010.

[6] E. Bingham, A. Gionis, N. Haiminen, H. Hiisilä, H. Mannila, and E. Terzi. Segmentation and dimensionality reduction. In *SDM*, 2006.

[7] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer-Verlag, 2nd edition, 1991.

[8] K. Chakrabarti, E. Keogh, S. Mehotra, and M. Pazzani. Localy adaptive dimensionality reduction for indexing large time series databases. *TODS*, 27(2), 2002.

[9] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *VLDB*, 2002.

[10] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *KDD*, 2003.

[11] N. M. A. Christoforos Anagnostopoulos and D. J. Hand. Streaming covariance selection with applications to adaptive querying in sensor networks. *The Computer Journal*, 53.

[12] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *KDD*, 1998.

[13] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD*, 2004.

[14] K. I. Diamantaras and S.-Y. Kung. *Principal Component Neural Networks: Theory and Applications*. John Wiley, 1996.

[15] P. Domingos and G. Hulten. Mining high-speed data streams. In *KDD*, 2000.

[16] D. Donoho. Compressed sensing. *IEEE TOIT*, 52, 2006.

[17] M. G. Elefky, W. G. Aref, and A. K. Elmagarmid. Using convolution to mine obscure periodic patterns in one pass. In *EDBT*, 2004.

[18] M. G. Elefky, W. G. Aref, and A. K. Elmagarmid. WARP: Time warping for periodicity detection. In *ICDM*, 2005.

[19] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, 1994.

[20] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[21] S. Gandhi, S. Nath, S. Suri, and J. Liu. GAMPS: Compressing multi sensor data by grouping and amplitude scaling. In *SIGMOD*, 2009.

[22] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining data streams under block evolution. *SIGKDD Explorations*, 3(2):1–10, 2002.

[23] M. Ghil, M. R. Allen, M. D. Dettinger, K. Ide, D. Kondrashov, M. E. Mann, A. W. Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Yiou. Advanced spectral methods for climatic time series. *Rev. Geophys.*, 40(1), 2002.

[24] S. Guha, D. Gunopulos, and N. Koudas. Correlating synchronous and asynchronous data streams. In *KDD*, 2003.

[25] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE TKDE*, 15(3):515–528, 2003.

[26] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. *IEEE TOIT*, 52, 2006.

[27] S. Haykin. *Adaptive Filter Theory.* Prentice Hall, 1992.

[28] M.-J. Hsieh, M.-S. Chen, and P. S. Yu. Integrating DCT and DWT for approximating cube streams. In *CIKM*, 2005.

[29] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *KDD*, 2001.

[30] A. Hyvärinen, P. Hoyer, and E. Oja. Sparse code shrinkage for image denoising. In *WCCI*, 1998.

[31] T. Idé and K. Inoue. Knowledge discovery from heterogeneous dynamic systems using change-point correlations. In *SDM*, 2005.

[32] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *VLDB*, 2000.

[33] I. T. Jolliffe. *Principal Component Analysis.* Springer, 2002.

[34] T. Kailath. *Linear Systems.* Prentice Hall, 1980.

[35] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD*, 2004.

[36] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT*, 2004.

[37] S. Lin, D. Gunopulos, V. Kalogeraki, and S. Lonardi. A data compression technique for sensor networks with dynamic bandwidth allocation. In *TIME*, 2005.

[38] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *ICDE*, 2005.

[39] A. Mueen, S. Nath, and J. Liu. Fast approximate correlation for massive time-series data. In *SIGMOD*, 2010.

[40] D. D. Muresan and T. W. Parks. Adaptive principal components and image denoising. In *ICIP*, 2003.

[41] E. Oja. Neural networks, principal components, and subspaces. *Intl. J. Neural Syst.*, 1:61–68, 1989.

[42] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos. Streaming time series summarization using user-defined amnesic functions. *IEEE TKDE*, 20, 2008.

[43] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, unsupervised stream mining. *VLDB J.*, 13(3), 2004.

[44] S. Papadimitriou, J. Sun, and C. Faloutsos. SPIRIT: Streaming pattern discovery in multiple time-series. In *VLDB*, 2005.

[45] P. Patel, E. Keogh, J. Lin, and S. Lonardi. Mining motifs in massive time series databases. In *ICDM*, 2002.

[46] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge Univ. Press, 2000.

[47] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *ICDM*, 2011.

[48] G. Reeves, J. Liu, S. Nath, and F. Zhao. Managing massive time series streams with multi-scale compressed trickles. In *VLDB*, 2009.

[49] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. BRAID: Stream mining through group lag correlations. In *SIGMOD*, 2005.

[50] J. Shieh and E. J. Keogh. Polishing the right apple: Anytime classification also benefits data streams with constant arrival times. In *ICDM*, 2010.

[51] T. A. Tarik Arici and Y. Altunbasak. A prediction error-based hypothesis testing method for sensor data acquisition.

[52] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *J. Royal Stat. Soc. B*, 61(3), 1999.

[53] K. Ueno, X. Xi, E. J. Keogh, and D.-J. Lee. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In *ICDM*, 2006.

[54] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD*, 2003.

[55] L. Wei, E. J. Keogh, H. V. Herle, and A. Mafra-Neto. Atomic wedgie: Efficient query filtering for streaming times series. In *ICDM*, 2005.

[56] B. Yang. Projection approximation subspace tracking. *IEEE Trans. Sig. Proc.*, 43(1):95–107, 1995.

[57] S. S. Yann-Aël Le Borgne and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks.

[58] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE*, 2000.

[59] P. Yiou, D. Sornette, and M. Ghil. Data-adaptive wavelets and multi-scale singular-spectrum analysis. *Physica D*, 142, 2000.

[60] P. Young. *Recursive Estimation and Time-Series Analysis: An Introduction*. Springer-Verlag, 1984.

[61] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *SIGMOD*, 1996.

[62] Y. Zhu and D. Shasha. StatStream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, 2002.

[63] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, Incremental tensor analysis: Theory and application. In *TKDD*, 2008.

Chapter 6

MINING SENSOR DATA STREAMS

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY
charu@us.ibm.com

Abstract

In recent years, advances in hardware technology have facilitated new ways of collecting data continuously. One such application is that of sensor data, which may continuously monitor large amounts of data for storage and processing. In this paper, we will discuss the general issues which arise in mining large amounts of sensor data. In many cases, the data patterns may evolve continuously, as a result of which it is necessary to design the mining algorithms effectively in order to account for changes in underlying structure of the data stream. This makes the solutions of the underlying problems even more difficult from an algorithmic and computational point of view. In this chapter we will provide an overview of the problem of data stream mining and the unique challenges that data stream mining poses to different kinds of sensor applications.

Keywords: Data Streams, Sensor Data, Sensor Stream Mining

1. Introduction

In recent years, advances in sensor technology have lead to the ability to collect large amounts of data from sensors in an automated way. When the volume of the underlying data is very large, it leads to a number of computational and mining challenges:

- With increasing volume of the data, it is no longer possible to process the data efficiently by using multiple passes. Rather, one

can process a data item at most once. This leads to constraints on the implementation of the underlying algorithms. Therefore, stream mining algorithms typically need to be designed so that the algorithms work with one pass of the data.

- In most cases, there is an inherent temporal component to the stream mining process. This is because the data may evolve over time. This behavior of data streams is referred to as *temporal locality*. Therefore, a straightforward adaptation of one-pass mining algorithms may not be an effective solution to the task. Stream mining algorithms need to be carefully designed with a clear focus on the evolution of the underlying data.

- Data which is collected from sensors is often uncertain and error prone, as a result of which it is critical to be able to reduce the effects of the uncertainty in the mining process.

Another important characteristic of sensor data streams is that they are often mined in a distributed fashion. In some cases, intermediate sensor nodes may have limited processing power, and it may be desirable to perform in-network sensor processing for a variety of mining applications. In such cases, the application algorithms need to be designed with such criteria in mind [30, 60]. This chapter will provide an overview of the key challenges in sensor stream mining algorithms which arise from the unique setup in which these problems are encountered.

This chapter is organized as follows. In the next section, we will discuss the generic challenges which arise in the context of storage and processing of sensor data. The next section deals with several issues which arise in the context of data stream management. In section 3, we discuss several mining algorithms on the data stream model. Section 4 discusses various scientific applications of data streams. Section 5 discusses the research directions and conclusions.

2. Sensor Stream Mining Issues

Since data streams are processes which create large volumes of incoming data, they lead to several challenges in both processing the data as well as applying traditional database operations. Therefore, new designs of data streaming systems are required for handling sensor data [23]. The challenging issues in sensor stream mining may arise during different phases including data collection, transmission, storage and processing. Some of the these key issues are as follows:

2.1 Data Uncertainty and Volume

Typical sensor mining applications collect large amounts of data, which is often subject to uncertainty and errors. This is because sensors often have errors in data collection and transmission. In many cases, where the battery runs out, the data may also be incomplete. Therefore, methods are required to store and process the uncertainty in the underlying data. A common technique is to perform *model driven data acquisition* [36], which explicitly models the uncertainty during the acquisition process. Furthermore, new methods must be constructed in order to explicitly use these uncertainty models during data processing. A detailed discussion of a variety of methods for modeling and mining uncertain data are proposed in [14].

A variety of techniques may be used in order to handle the large volume of sensor data. One method may be to simply lower the sampling rate for capturing or transmitting the data. This reduces the granularity of the data collection process. Many techniques have also been proposed [34, 35] in order to reduce or compress the volume of the data in the sensor network. Another approach is to discard parts of the data even after collection and transmission. For example, when the incoming rate of the data streams is higher than that can be processed by the system, techniques are required in order to selectively pick data points from the stream, without losing accuracy. This technique is known as *loadshedding*. Since sensor generated data streams are generated by processes which are extraneous to the stream processing application, it is not possible to control the incoming stream rate. As a result, it is necessary for the system to have the ability to quickly adjust to varying incoming stream processing rates. One particular type of adaptivity is the ability to gracefully degrade performance via "load shedding" (dropping unprocessed tuples to reduce system load) when the demands placed on the system cannot be met in full given available resources. The loadshedding can be tailored to specific kinds of applications such as query processing or data mining. A discussion of several loadshedding techniques are provided in [4]. Finally, a method for reducing the data volume is that of sensor selection in which data from only a particular set of sensors is transmitted at a particular time, so that most of the essential information is retained. This is also useful for reducing the power requirements of transmission. We will discuss this method in the next subsection.

2.2 Power Issues in Sensor Collection and Transmission

Since sensor nodes have limited battery power, the issue of data collection and transmission is a challenging one for sensor networks. Sensor nodes have limited battery power, as a result of which it is necessary to collect and transmit the data on a limited basis. This means that it is often necessary to collect only a subset of the data for mining purposes. A classic technique which is often used in order to reduce the amount of collected data is *sensor selection*. In data driven sensor selection, goal-oriented techniques are used in order to reduce the amount of data collected for mining purposes [19, 2–1, 45, 61]. In most of these methods, the idea is to use the massive redundancy across the different sensors in order to reduce the total data which is collected. For example, in an environmental monitoring applications, two sensors which are located physically close together may often have very similar readings. In such cases, the correlations across the different sensors are leveraged in order to select a small number of sensors from which the data is collected. The values on the other sensors can be predicted from this small set. We note that this small set may vary over time, as different sensors may be inoperative at different times, or the correlations among the data may change. Methods for power-efficient and dynamic sensor selection are discussed in [1]. Another technique which is often used in order to reduce the power transmission costs is a method referred to as *in network* processing. We will discuss this technique in the next subsection.

2.3 In-Network Processing

Since sensor networks may use hundreds and thousands of nodes over a large area, and all the data from the different nodes may be need to be fused, this can incur significant communication costs, of all the raw data is directly transmitted to a central server for processing. While such a naive solution is easy to implement, its energy costs may be too large to make it practical for very large scale sensor networks which are distributed over wide regions. Since the cost of transmission is higher than computation, it is usually advantageous to organize the sensors into clusters. In such an environment, the data gathered by the sensors is *processed within the network* and only aggregated information is returned to the central location. This responsibility is typically provided to certain nodes in the network, which are referred to as *aggregators*. Numerous functions can be designed for such nodes, and the corresponding data sent may depend upon the relevant aggregate queries which are posed at the central server. Thus, the underlying assumption is that such em-

bedded devices in the network are smart enough to be able to compute functions of modest complexity. Such an approach results in a reduction of the transmission costs, both because of the smaller distances of transmission in a clustered environment, and also because only the aggregated data is transmitted (which has much lower volume than the raw data). It is possible to design different kinds of cluster hierarchies in order to optimize the transmission costs in the underlying network. A detailed discussion of the different aspects of in-network query processing may be found in [63, 78].

3. Stream Mining Algorithms

In this section, we will discuss the key stream mining problems and will discuss the challenges associated with each problem. We will also provide a broad overview of the different directions of research for these problems.

3.1 Data Stream Clustering

Clustering is a widely studied problem in the data mining literature. However, it is more difficult to adapt arbitrary clustering algorithms to data streams because of one-pass constraints on the data set. An interesting adaptation of the k-means algorithm has been discussed in [43] which uses a partitioning based approach on the entire data set. This approach uses an adaptation of a k-means technique in order to create clusters over the entire data stream. However, in practical applications, it is often desirable to be able to examine clusters over user-specified time-horizons. For example, an analyst may desire to examine the behavior of the clusters in the data stream over the past one week, the past one month, or the past year. In such cases, it is desirable to store *intermediate cluster statistics*, so that it is possible to leverage these in order to examine the behavior of the underlying data.

One such technique is micro-clustering [10], in which we use cluster feature vectors [81] in order to perform stream clustering. The cluster feature vectors keep track of the first-order and second-order moments of the underlying data in order to perform the clustering. These features satisfy the following critical properties which are relevant to the stream clustering process:

- **Additivity Property:** The statistics such as the first- or second-order moments can be maintained as a simple addition of statistics over data points. This is critical in being able to maintain the statistics efficiently over a fast data stream. Furthermore, additivity also implies subtractivity; thus, it is possible to obtain the

statistics over a particular time horizon, by subtracting out the statistics at the beginning of the horizon from the statistics at the end of the horizon.

- **Computational Convenience:** The first and second order statistics can be used to compute a vast array of cluster parameters such as the cluster centroid and radius. This is useful in order to be able to compute important cluster characteristics in real time.

It has been shown in [10], that the micro-cluster technique is much more effective and versatile than the k-means based stream technique discussed in [43]. This broad technique has also been extended to a variety of other kinds of data. Some examples of such data are as follows:

- **High Dimensional Data:** The stream clustering method can also be extended to the concept of projected clustering [5]. A technique for high dimensional projected clustering of data streams is discussed in [11]. In this case, the same micro-cluster statistics are used for maintaining the characteristics of the clusters, except that we also maintain additional information which keeps track of the projected dimensions in each cluster. The projected dimensions can be used in conjunction with the cluster statistics to compute the projected distances which are required for intermediate computations. Another innovation proposed in [11] is the use of decay-based approach for clustering. The idea in the decay-based approach is relevant for the case of evolving data stream model, and is applicable not just to the high dimensional case, but any of the above variants of the micro-cluster model. In this approach, the weight of a data point is defined as $2^{-\lambda \cdot t}$, where t is the current time-instant. Thus, each data point has a half-life of $1/\lambda$, which is the time in which the weight of the data point reduces by a factor of 2. We note that the decay-based approach poses a challenge because the micro-cluster statistics are affected at each clock tick, even if no points arrive from the data stream. In order to deal with this problem, a *lazy approach* is applied to decay-based updates, in which we update the decay-behavior for a micro-cluster only if a data point is added to it. The idea is that as long as we keep track of the last time t_s at which the micro-cluster was updated, we only need to multiply the micro-cluster statistics by $2^{-\lambda(t_c - t_s)}$, where t_c is the current time instant. After multiply the decay statistics by this factor, it is possible to add the micro-cluster statistics of the current data point. This approach can be used since the statistics of each micro-cluster decay by the same factor in each track, and it is therefore possible to implicitly keep track of the decayed values,

as long as a data point is not added to the micro-cluster. In the latter case, the statistics need to be updated explicitly, while other counts can still be maintained implicitly.

- **Uncertain Data:** In many cases, such as in sensor networks, the underlying data may be noisy and uncertain. In such cases, it may be desirable to incorporate the uncertainty into the clustering process. In order to do so, the micro-cluster statistics are appended with the information about the underlying uncertainty in the data. This information can be used in order to make more robust clustering computations. The advantages of using the uncertainty into the clustering process are illustrated in [7].
- **Text and Categorical Data:** A closely related problem is that of text and categorical data. The main difference with the quantitative domain is the nature of the statistics which are stored for clustering purposes. In this case, we maintain the counts of the frequencies of the discrete attributes in each cluster. Furthermore, we also maintain the inter-attribute correlation counts which may be required in a variety of applications. In [12], an efficient algorithm has been proposed for clustering text and categorical data streams. This algorithm also allows for a decay-based approach as in [11]. Text and categorical streams often arise in the context of social sensors such as micro-blogging sites, or other tagging or event detection scenarios.

In addition, a number of density-based methods [25, 28] have also been proposed for the problem of stream clustering.

In the context of sensor networks, the stream data is often available only in a *distributed setting*, in which large volumes of data are collected separately at the different sensors. A natural approach for clustering such data is to transmit all of the data to a centralized server. The clustering can then be performed at the centralized server in order to determine the final results. Unfortunately, such an approach is extremely expensive in terms of its communication costs. Therefore, it is important to design a method which can reduce the communication costs among the different processors. A method proposed in [32] performs local clustering at each node, and merges these different clusters into a single global clustering at low communication cost. Two different methods are proposed in this work. The first method determines the cluster centers by using a *furthest point algorithm*, on the current set of data points at the local site. In the furthest point algorithm, the center of a cluster is picked as a furthest point to the current set of centers. For any incoming data point, it is assigned to its closest center, as long

the distance is within a certain factor of an optimally computed radius. Otherwise, a re-clustering is forced by applying the furthest point algorithm on current set of points. After the application of the furthest point algorithm, the centers are transmitted to the central server, which then computes a global clustering from these local centers over the different nodes. These global centers can ten be transmitted to the local nodes if desired. One attractive feature of the method is that an approximation bound is proposed on the quality of the clustering. A second method for distributed clustering proposed in [32] is the *parallel guessing algorithm*. Another method for distributed sensor stream clustering which reduces the dimensionality and communication cost by maintaining an online discretization may be found in [68].

3.2 Data Stream Classification

The problem of classification is perhaps one of the most widely studied in the context of data stream mining. The problem of classification is made more difficult by the evolution of the underlying data stream. Therefore, effective algorithms need to be designed in order to take temporal locality into account. The concept of stream evolution is sometimes referred to as *concept drift* in the stream classification literature. Some of these algorithms are designed to be purely one-pass adaptations of conventional classification algorithms [39], whereas others (such as the methods in [13, 48]) are more effective in accounting for the evolution of the underlying data stream. The broad methods which are studied for classification in the data stream scenario are as follows:

VFDT Method: The VFDT (Very Fast Decision Trees) method has been adapted to create decision trees which are similar to those constructed by a conventional learner with the use of sampling based approximations. The VFDT method splits a tree using the current best attribute, taking into consideration the fact that the number of examples used are sufficient to preserve the Hoeffding bound in a way that the output is similar to that of a conventional learner. The key question during the construction of the decision tree is the choice of attributes to be used for splits. Approximate ties are broken using a user-specified threshold of acceptable error-measure for the output. It can be shown that for any small value of δ, a particular choice of the split variable is the correct choice with probability at least $1 - \delta$, if a sufficient number of stream records have been processed. This number has been shown in [39] to increase at a relatively modest rate of $\ln(1/\delta)$. This bound can then be extended to the entire decision tree, so as to quantify the probability that the same decision tree as a conventional learner is cre-

ated. The VFDT method has also been extended to the case of evolving data streams. This framework is referred to as CVFDT [48], and it runs VFDT over fixed sliding windows in order to always have the most updated classifier. Jin and Agrawal [50] have extended the VFDT algorithm in order to process numerical attributes and reduce the sample size which is calculated using the Hoeffding bound. Since this approach reduces the sample size, it improves efficiency and space requirements for a given level of accuracy.

On Demand Classification: While most stream classification methods are focussed on a training stream, the on demand method is focussed on the case when both the training and the testing stream evolves over time. In the on demand classification method [13], we create class-specific micro-clusters from the underlying data. For an incoming record in the test stream, the class label of the closest micro-cluster is used in order to determine the class label of the test instance. In order to handle the problem of stream evolution, the micro-clusters from the specific time-horizon are used for the classification process. A key issue in this method is the choice of horizon which should be used in order to obtain the best classification accuracy. In order to determine the best horizon, a portion of the training stream is separated out and the accuracy is tested over this portion with different horizons. The optimal horizon is then used in order to classify the test instance.

Ensemble-based Classification: This technique [74] uses an ensemble of classification methods such as C4.5, RIPPER and naive Bayes in order to increase the accuracy of the predicted output. The broad idea is that a data stream may evolve over time, and a different classifier may work best for a given time period. Therefore, the use of an ensemble method provides robustness in the concept-drifting case.

Compression-based Methods: An interesting method for real-time classification of streaming sensor data with the use of compression techniques has been proposed in [57]. In this approach, time-series bitmaps, which can be updated in constant time are used as efficient classifiers. Because of the ability of be updated in constant time, these classifiers are very efficient in practice. The effectiveness of this approach has been illustrated on a number of insect-tracking data sets.

In the context of sensor networks, data streams may often have a significant level of errors and uncertainty. Data uncertainty brings a number of unique challenges with it in terms of the determination of the important features to be used for the classification process. In this context, a number of algorithms have been proposed for classification of uncertain data streams [14, 15]. In particular, the method discussed in

[15] constructs a density-based framework to summarize the uncertain data stream effectively and use it for classification purposes.

3.3 Frequent Pattern Mining

The problem of frequent pattern mining was first introduced in [16], and was extensively analyzed for the conventional case of disk resident data sets. In the case of data streams, one may wish to find the frequent itemsets either over a sliding window or the entire data stream [44, 53]. In the case of data streams, the problem of frequent pattern mining can be studied under several models:

Entire Data Stream Model: In this model, the frequent patterns need to be mined over the entire data stream. Thus, the main difference from a conventional pattern mining algorithm is that the frequent patterns need to be mined in one pass over the entire data stream. Most frequent pattern mining algorithms require multiple passes in order to estimate the frequency of patterns of different sizes in the data. A natural method for frequent pattern counting is to use sketch-based algorithms in order to determine frequent patterns. Sketches are often used in order to determine heavy-hitters in data streams, and therefore, an extension of the methodology to the problem of finding frequent patterns is natural. Along this line, Manku and Motwani [64] proposed the first one lass algorithm called *Lossy Counting*, in order to find all frequent itemsets over a data stream. The algorithm allows false positives, but not false negatives. Thus, for a given support level s, the algorithm is guaranteed not to contain all frequent itemsets whose support is greater than $s - \epsilon$. Another interesting approach in [80] determines all the frequent patterns whose support is greater than s with probability at least $1 - \delta$, which the value of δ is as small as desired, as long as one is willing to add space and time complexity proportional to $\ln(1/\delta)$.Thus, this model does not allow false negatives, but may miss some of the frequent patterns. The main advantage of such a technique is that it is possible to provide a more concise set of frequent patterns at the expense of losing some of the patterns with some probability which is quite low for practical purposes.

Sliding Window Model: In many cases, the data stream may evolve over time, as a result of which it is desirable to determine all the frequent patterns over a particular sliding window. A method for determining the frequent patterns over a sliding window is discussed in [29]. The main assumption of this approach is that the number of frequent patterns are not very large, and therefore, it is possible to hold the transactions in each sliding window in main memory. The main focus of this approach is to determine closed frequent itemsets over the data stream. A new

mining algorithm called MOMENT is proposed, and the main idea is based on the fact that the boundary between closed frequent itemsets and frequent itemsets moves very slowly. A closed enumeration tree is developed in order to keep track of the boundary between closed frequent itemsets and the rest of the itemsets. Another method which is able to mine frequent itemsets over arbitrary time granularities is referred to as FPSTREAM [42]. This method is essentially an adaptation of the FP-Tree method to data streams.

Damped Window Model: We note that pure sliding windows are not the only way by which the evolution of data streams can be taken into account during the mining process. A second way is to introduce a *decay factor* into the computation. Specifically, the weight of each transaction is multiplied by a factor of $f < 1$, when a new transaction arrives. The overall effect of such an approach is to create an exponential decay function on the arrivals in the data stream. Such a model is quite effective for evolving data stream, since recent transactions are counted more significantly during the mining process. An algorithm proposed in [27] maintains a lattice for recording the potentially frequent itemsets and their counts. While the counts of each lattice may change upon the arrival of each transaction, a key observation is that it is sufficient to update the counts in a lazy way. Specifically, the decay factor is applied only to those itemsets whose counts are affected by the current transaction. However, the decay factor will have to be applied in a modified way by taking into account the last time that the itemset was touched by an update. In other words, if t_c be the current transaction index, and the last time the count for the itemset was updated was at transaction index $t_s < t_c$, then we need to multiply the current counts of that itemset by $f^{t_s - t_c}$ before incrementing the count of this modified value. This approach works because the counts of each itemset reduce by the same decay factor in each iteration, as long as a transaction count is not added to it. We note that such a lazy approach is also applicable to other mining problems, where statistics are represented as the sum of decaying values. For example, in [11], a similar lazy approach is used in order to maintain decay-based micro-cluster statistics for a high dimensional projected stream clustering algorithm.

3.4 Change Detection in Data Streams

As discussed earlier, the patterns in a data stream may evolve over time. In many cases, it is desirable to track and analyze the nature of these changes over time. In [8, 37, 59], a number of methods have been discussed for change detection of data streams. In addition, data

stream evolution can also affect the behavior of the underlying data mining algorithms since the results can become stale over time. The broad algorithms for change diagnosis in data streams are as follows:

Velocity Density Estimation: In velocity density estimation [8], we compute the rate of change of data density of different points in the data stream over time. Depending upon the direction of density rate of change, one may identify regions of *dissolution*, *coagulation* and *shift*. Spatial profiles can also be constructed in order to determine the directions of shift in the underlying data. In addition, it is possible to use the velocity density concept in order to identify those combinations of dimensions which have a high level of evolution. Another technique for change quantification is discussed in [37], which uses methods for probability difference quantification in order to identify the changes in the underlying data. In [59], a method is discussed in order to determine statistical changes in the underlying data. Clustering [10] can be used in order to determine significant evolution in the underlying data. In [10], micro-clustering is used in order to determine significant clusters which have evolved in the underlying data.

A separate line of work is the determination of significant changes in the results of data mining algorithms because of evolution. For example in [10], it has been shown how to determine significant evolving clusters in the underlying data. In [13], a similar technique has been used to keep a refreshed classification model in the presence of evolving data. In this respect, micro-clustering provides an effective technique, since it provides a way to store intermediate statistics of the underlying data in the form of clusters. In [13], a micro-cluster based nearest neighbor classifier is used in order to classify evolving data streams. The key idea is to construct class-specific micro-clusters over a variety of time horizons, and then utilize the time horizon with the greatest accuracy in order to perform the classification process. The issue of stream evolution has been extended to many other problems such as synopsis construction and reservoir sampling [6]. We will discuss some of the synopsis construction methods later.

3.5 Synopsis Construction in Data Streams

The large volume of data streams poses unique space and time constraints on the computation process. Many query processing, database operations, and mining algorithms require efficient execution which can be difficult to achieve with a fast data stream. Furthermore, since it is impossible to fit the entire data stream within the available space, the space efficiency of the approach is a major concern. In many cases, it

may be acceptable to generate *approximate solutions* for many problems by summarizing the data in a time and space-efficient way. In recent years a number of *synopsis structures* have been developed, which can be used in conjunction with a variety of mining and query processing techniques [41]. Some key synopsis methods include those of sampling, wavelets, sketches and histograms. The key challenges which arise in the context of synopsis construction of data streams are as follows:

Broad Applicability: The synopsis structure is typically used as an intermediate representation, which is then leveraged for a variety of data mining and data management problems. Therefore, the synopsis structure should be c0onstructed in such a way that it has applicability across a wide range of problems.

One-pass constraint: As in all data stream algorithms, the one-pass constraint is critical to synopsis construction algorithms. We would like to design all synopsis construction algorithms in one pass, and this is not the case for most traditional methods. In fact, even simply methods such as sampling need to be re-designed in order to handle the one-pass constraint.

Time and Space Efficiency: Since data streams have a very large volume, it is essential to create the synopsis in a time- and space-efficient way. In this sense, some of the probabilistic techniques such as sketches are extremely effective for counting-based applications, since they require constant-space for provable probabilistic accuracy. In other words, the time- and space-efficiency depends only upon the accuracy of the approach rather than the length of the data stream.

Data Stream Evolution: Since the stream evolves over time, a synopsis structure which is constructed from the overall behavior of the data stream is not quite as effective as one which uses recent history. Consequently, it is often more effective to create synopsis structures which either work with sliding windows, or use some decay-based approach in order to weight the data stream points.

One key characteristic of many of the above methods is that while they work effectively in the 1-dimensional case, they often lose their effectiveness in the multi-dimensional case either because of data sparsity or because of inter-attribute correlations. Next, we will discuss the broad classes of techniques which are used for synopsis construction in data streams. Each of these techniques have their own advantages in different scenarios, and we will take care to provide an overview of the different array of methods which are used for synopsis construction in data streams. The broad techniques which are used for synopsis construction in data streams are as follows:

Reservoir Sampling: Sampling methods are widely used for tradi-

tional database applications, and are extremely popular because of their broad applicability across a wide array of tasks in data streams. A further advantage of sampling methods is that unlike many other synopsis construction methods, they maintain their inter-attribute correlations across samples of the data. It is also often possible to use probabilistic inequalities in order to bound the effectiveness of a variety of applications with sampling methods.

However, a key problem in extending sampling methods to the data stream scenario, is that one does not know the total number of data points to be sampled in advance. Rather, one must maintain the sample in a dynamic way over the entire course of the computation. A method called reservoir sampling was first proposed in [72], which maintains such a sample dynamically. This technique was originally proposed in the context of one-pass access of data from magnetic-storage devices. However, the techniques also naturally extend to the data stream scenario.

Let us consider the case, where we wish to obtain an unbiased sample of size n from the data stream. In order to initialize the approach, we simply add the first n points from the stream to the reservoir. Subsequently, when the $(t+1)$th point is received, it is added to the reservoir with probability $n/(t+1)$. When the data point is added to the reservoir, it replaces a random point from the reservoir. It can be shown that this simple approach maintains the uniform sampling distribution from the data stream. We note that the uniform sampling approach may not be very effective in cases where the data stream evolves significantly. In such cases, one may either choose to generate the stream sample over a sliding window, or use a decay-based approach in order to bias the sample. An approach for sliding window computation over data streams is discussed in [20].

A second approach [6] uses biased decay functions in order to construct synopsis from data streams. It has been shown in [6] that the problem is extremely difficult for arbitrary decay functions. In such cases, there is no known solution to the problem. However, it is possible to design very simple algorithms for some important classes of decay functions. One of these classes of decay functions is the *exponential decay function*. The exponential decay function is extremely important because of its *memory less property*, which guarantees that the future treatment of a data point is independent of the past data points which have arrived. An interesting result is that by making simple implementation modifications to the algorithm of [72] in terms of modifying the probabilities of insertion and deletion, it is possible to construct a robust algorithm for this problem. It has been shown in [6] that the approach is quite

effective in practice, especially when there is significant evolution of the underlying data stream.

While sampling has several advantages in terms of simplicity and preservation of multi-dimensional correlations, it loses its effectiveness in the presence of data sparsity. For example, a query which contains very few data points is unlikely to be accurate with the use of a sampling approach. However, this is a general problem with most techniques which are effective at counting frequent elements, but are not quite as effective at counting rare or distinct elements in the data stream.

Sketches: Sketches use some properties of random sampling in order to perform counting tasks in data streams. Sketches are most useful when the *domain size* of a data stream is very large. In such cases, the number of possible distinct elements become very large, and it is no longer possible to track them in space-constrained scenarios. There are two broad classes of sketches: *projection based* and *hash based*. We will discuss each of them in turn.

Projection based sketches are constructed on the broad idea of random projection [54]. The most well known projection-based sketch is the AMS sketch [17, 18], which we will discuss below. It has been shown in [54], that by by randomly sampling subspaces from multi-dimensional data, it is possible to compute ϵ-accurate projections of the data with high probability. This broad idea can easily be extended to the massive domain case, by viewing each distinct item as a dimension, and the counts on these items as the corresponding values. The main problem is that the vector for performing the projection cannot be maintained explicitly since the length of such a vector would be of the same size as the number of distinct elements. In fact, since the sketch-based method is most relevant in the distinct element scenario, such an approach defeats the purpose of keeping a synopsis structure in the first place.

Let us assume that the random projection is performed using k sketch vectors, and r_i^j represents the jth vector for the ith item in the domain being tracked. In order to achieve the goal of efficient synopsis construction, we store the random vectors implicitly in the form of a seed, and this can be used to dynamically generate the vector. The main idea discussed in [49] is that it is possible to generate random vectors with a seed of size $O(\log(N))$, provided that one is willing to work with the restriction that $r_i^j \in \{-1, +1\}$ should be 4-wise independent. The sketch is computed by adding r_i^j to the jth component of the sketch for the ith item. In the event that the incoming item has frequency f, we add the value $f \cdot r_i^j$. Let us assume that there are a total of k sketch components which are denoted by $(s_1 \ldots s_k)$. Some key properties of the pseudo-

random number generator approach and the sketch representation are as follows:

- A given component r_i^j can be generated in poly-logarithmic time from the seed. The time for generating the seed is poly-logarithmic in the domain size of the underlying data.
- A variety of approximate aggregate functions on the original data can be computed using the sketches.

Some example of functions which can be computed from the sketch components are as follows:

- **Dot Product of two streams:** If $(s_1 \ldots s_k)$ be the sketches from one stream, and $(t_1 \ldots t_k)$ be the sketches from the other stream, then $s_j cdott_j$ is a random variable whose expected value of the dot product.
- **Second Moment:** If $(s_1 \ldots s_k)$ be the sketch components for a data stream, it can be shown that the expected value of s_j^2 is the second moment. Furthermore, by using Chernoff bounds, it can be shown that by selecting the median of $O(\log(1/\delta)$ averages of $O(1/\epsilon^2)$ copies of $s_j]cdott_j$, it is possible to guarantee the accuracy of the approximation to within $1^{\pm}\epsilon$ with probability at least $1-\delta$.
- **Frequent Items:** The frequency of the ith item in the data stream is computed by by multiplying the sketch component s_j by r_i^j. However, this estimation is accurate only for the case of frequent items, since the error is estimation is proportional to the overall frequency of the items in the data stream.

More details of computations which one can perform with the AMS sketch are discussed in [17, 18].

The second kind of sketch which is used for counting is the *count-min* sketch [31]. The count-min sketch is based upon the concept of hashing, and uses $k = \ln(1/\delta)$ pairwise-independent hash functions, which hash onto integers in the range $(0 \ldots e/\epsilon)$. For each incoming item, the k hash functions are applied and the frequency count is incremented by 1. In the event that the incoming item has frequency f, then the corresponding frequency count is incremented by f. Note that by hashing an item into the k cells, we are ensuring that we maintain an overestimate on the corresponding frequency. It can be shown that the minimum of these cells provides the ϵ-accurate estimate to the frequency with probability at least $1-\delta$. It has been shown in [31] that the method can also be naturally extended to other problems such as finding the dot product

Table 6.1. An Example of Wavelet Coefficient Computation

Granularity (Order k)	**Averages** Φ values	**DWT Coefficients** ψ values
$k = 4$	(8, 6, 2, 3, 4, 6, 6, 5)	-
$k = 3$	(7, 2.5, 5, 5.5)	(1, -0.5,-1, 0.5)
$k = 2$	(4.75, 5.25)	(2.25, -0.25)
$k = 1$	(5)	(-0.25)

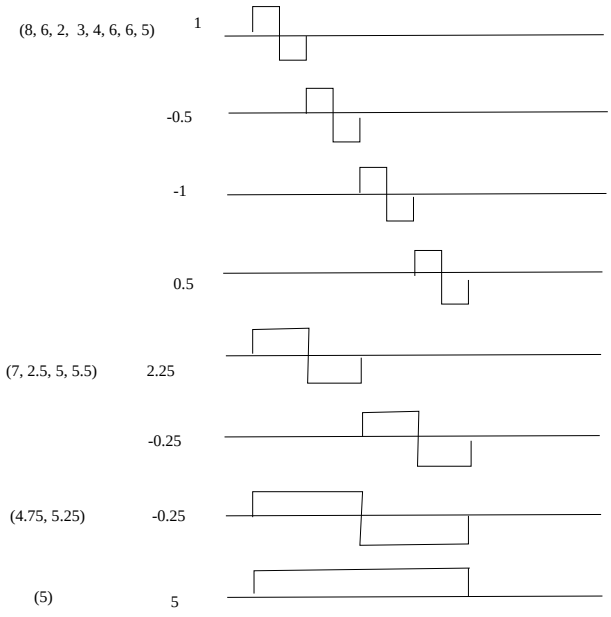


Figure 6.1. Illustration of the Wavelet Decomposition

or the second-order moments. The count-min sketch is typically more effective for problems such as frequency-estimation of individual items than the projection-based AMS sketch. However, the AMS sketch is more effective for problems such as second-moment estimation.

Wavelet Decomposition: Another widely known synopsis representation in data stream computation is that of the wavelet representation. One of the most widely used representations is the *Haar Wavelet*. We will discuss this technique in detail in this section.

This technique is particularly simple to implement, and is widely used in the literature for hierarchical decomposition and summarization. The basic idea in the wavelet technique is to create a decomposition of the data characteristics into a set of wavelet functions and basis functions. The property of the wavelet method is that the higher order coefficients

of the decomposition illustrate the broad trends in the data, whereas the more localized trends are captured by the lower order coefficients.

We assume for ease in description that the length q of the series is a power of 2. This is without loss of generality, because it is always possible to decompose a series into segments, each of which has a length that is a power of two. The Haar Wavelet decomposition defines 2^{k-1} coefficients of order k. Each of these 2^{k-1} coefficients corresponds to a contiguous portion of the time series of length $q/2^{k-1}$. The ith of these 2^{k-1} coefficients corresponds to the segment in the series starting from position $(i-1) \cdot q/2^{k-1} + 1$ to position $i * q/2^{k-1}$. Let us denote this coefficient by ψ_k^i and the corresponding time series segment by S_k^i. At the same time, let us define the average value of the first half of the S_k^i by a_k^i and the second half by b_k^i. Then, the value of ψ_k^i is given by $(a_k^i - b_k^i)/2$. More formally, if Φ_k^i denote the average value of the S_k^i, then the value of ψ_k^i can be defined recursively as follows:

$$\psi_k^i = (\Phi_{k+1}^{2 \cdot i-1} - \Phi_{k+1}^{2 \cdot i})/2 \tag{6.1}$$

The set of Haar coefficients is defined by the Ψ_k^i coefficients of order 1 to $\log_2(q)$. In addition, the global average Φ_1^1 is required for the purpose of perfect reconstruction. We note that the coefficients of different order provide an understanding of the major trends in the data at a particular level of granularity. For example, the coefficient ψ_k^i is half the quantity by which the first half of the segment S_k^i is larger than the second half of the same segment. Since larger values of k correspond to geometrically reducing segment sizes, one can obtain an understanding of the basic trends at different levels of granularity. We note that this definition of the Haar wavelet makes it very easy to compute by a sequence of averaging and differencing operations. In Table 6.1, we have illustrated how the wavelet coefficients are computed for the case of the sequence $(8, 6, 2, 3, 4, 6, 6, 5)$. This decomposition is illustrated in graphical form in Figure 6.1. We also note that each value can be represented as a sum of $\log_2(8) = 3$ linear decomposition components. In general, the entire decomposition may be represented as a tree of depth 3, which represents the hierarchical decomposition of the entire series. This is also referred to as the *error tree*. In Figure 6.2, we have illustrated the error tree for the wavelet decomposition illustrated in Table 6.1. The nodes in the tree contain the values of the wavelet coefficients, except for a special *super-root* node which contains the series average. This super-root node is not necessary if we are only considering the relative values in the series, or the series values have been normalized so that the average is already zero. We further note that the number of wavelet coefficients in this series is 8, which is also the length of the original series. The original

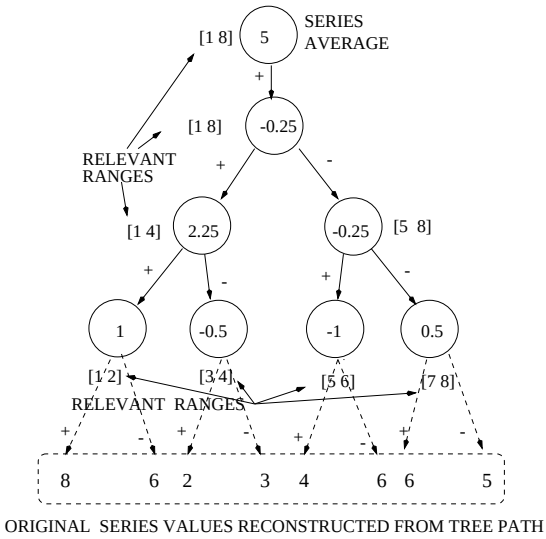


Figure 6.2. The Error Tree from the Wavelet Decomposition

series has been replicated just below the error-tree in Figure 6.2, and it can be reconstructed by adding or subtracting the values in the nodes along the path leading to that value. We note that each coefficient in a node should be added, if we use the left branch below it to reach to the series values. Otherwise, it should be subtracted. This natural decomposition means that an entire contiguous range along the series can be reconstructed by using only the portion of the error-tree which is relevant to it. Furthermore, we only need to retain those coefficients whose values are significantly large, and therefore affect the values of the underlying series. In general, we would like to minimize the reconstruction error by retaining only a fixed number of coefficients, as defined by the space constraints. While wavelet decomposition is easy to perform for multi-dimensional data sets, it is much more challenging for the case of data streams. This is because data streams impose a one-pass constraint on the wavelet construction process. A variety of one-pass algorithms for wavelet construction are discussed in [41].

Histograms: The technique of histogram construction is closely related to that of wavelets. In histograms the data is binned into a number of intervals along an attribute. For any given query, the counts from the bins can be utilized for query resolution. A simple representation of the histogram method would simply partition the data into equi-depth or equi-width intervals. The main inaccuracy with the use of histograms

is that the distribution of data points within a bucket is not retained, and is therefore assumed to be uniform. This causes inaccuracy because of extrapolation at the query boundaries. A natural choice is to use an equal number of counts in each bucket. This minimizes the error variation across different buckets. However, in the case of data streams, the boundaries to be used for equi-depth histogram construction are not known a-priori. We further note that the design of equi-depth buckets is exactly the problem of quantile estimation, since the equi-depth partitions define the quantiles in the data. Another choice of histogram construction is that of minimizing the variance of frequency variances of different values in the bucket. This ensures that the uniform distribution assumption is approximately held, when extrapolating the frequencies of the buckets at the two ends of a query. Such histograms are referred to as V-optimal histograms. Algorithms for V-optimal histogram construction are proposed in [51, 52]. A more detailed discussion of several algorithms for histogram construction may be found in [4].

3.6 Dimensionality Reduction and Forecasting in Data Streams

Because of the inherent temporal nature of data streams, the problems of dimensionality reduction and forecasting and particularly important. When there are a large number of simultaneous data stream, we can use the correlations between different data streams in order to make effective predictions [70, 75] on the future behavior of the data stream. In particular, the well known MUSCLES method [75] is useful in applying regression analysis to data streams. The regression analysis is helpful in predicting the future behavior of the data stream. A related technique is the SPIRIT algorithm, which explores the relationship between dimensionality reduction and forecasting in data streams. The primary idea is that a compact number of hidden variables can be used to comprehensively describe the data stream. This compact representation can also be used for effective forecasting of the data streams. A discussion of different dimensionality reduction and forecasting methods (including SPIRIT) is provided in [4].

3.7 Distributed Mining of Data Streams

In many instances, streams are generated at multiple distributed computing nodes. An example of such a case would be sensor networks in which the streams are generated at different sensor nodes. Analyzing and monitoring data in such environments requires data mining technology that requires optimization of a variety of criteria such as communication

costs across different nodes, as well as computational, memory or storage requirements at each node. There are several management and mining challenges in such cases. When the streams are collected with the use of sensors, one must take into account the limited storage, computational power, and battery life of sensor nodes. Furthermore, since the network may contain a very large number of sensor nodes, the effective aggregation of the streams becomes a considerable challenge. Furthermore, distributed streams also pose several challenges to mining problems, since one must integrate the results of the mining algorithms across different nodes. A detailed discussion of several distributed mining algorithms are provided in [4].

4. Sensor Applications of Stream Mining

Data streams have numerous applications in a variety of scientific scenarios. In this section, we will discuss different applications of data streams and how they tie in to the techniques discussed earlier.

4.1 Military Applications

Military applications typically collect large amounts of sensor data, for their use in a variety of applications such as the detection of events and anomalies in the data. Some classic examples of military applications are as follows:

4.1.1 Activity Monitoring. Military sensors are used for a variety of scenarios such as the detection of threats movements, sounds, or vibrations in the underlying data. For example, the movement of enemy tanks in a particular region may result in a particular combination of signals detected in the sound and activity sensors. Such monitoring may require the development of heterogeneous mining and fusion techniques [73], which can combine information from multiple sources in order to perform more effective mining. Such monitoring requires stream mining methods for the continuous detection of abnormalities, or for performing continuous queries in the underlying data [21, 22, 26, 66, 79].

4.1.2 Event Detection. This is related to the problem of activity monitoring, in that specific events are captured from the stream with the use of mining techniques. This requires the design of event detection algorithms from data streams. This typically requires the use of supervised learning algorithms in which the relationship of the events to the underlying stream attributes are learned from the training data. For example, such streams are quite common in social networks, in which

it is possible to determine the key events from the underlying social network from the patterns in the underlying text stream [11]. Other general methods for event detection in streams are discussed in [3, 65, 69, 76].

4.2 Cosmological Applications

In recent years, cosmological applications have created large volumes of data. The installation of large space stations, space telescopes and observatories result in large streams of data on different stars and clusters of galaxies. This data can be used in order to mine useful information about the behavior of different cosmological objects. Similarly, rovers and sensors on a planet or asteroid may send large amounts of image, video or audio data. In many cases, it may not be possible to manually monitor such data continuously. In such cases, it may be desirable to use stream mining techniques in order to detect the important underlying properties.

The amount of data received in a single day in such applications can often exceed several tera-bytes. These data sources are especially challenging since the underlying applications may be spatial in nature. In such cases, an attempt to compress the data using standard synopsis techniques may lose the structure of the underlying data. Furthermore, the data may often contain imprecision in measurements. Such imprecisions may result in the need for techniques which leverage the uncertainty information in the data in order to improve the accuracy of the underlying results.

4.3 Mobile Applications

Recently, new technologies have emerged which have allowed the construction of *wearable sensors* in the context of a variety of applications. For example, mobile phones carry a wide variety of sensors which can continuously transmit data that can be used for *social sensing* applications [62]. Similarly, wearable sensors have been designed for continuous monitoring in a wide variety of domains such as health-care [46, 71] or vehicular participatory sensing [47]. All vehicles which have been designed since the mid-nineties carry an *OBD Diagnostic System*, which collects a huge amount of information from the underlying vehicle operation. It is possible to use the information gleaned from on-board sensors in a vehicle in order to monitor the diagnostic health of the vehicle as well as driver characterization. Another well known method is the *VEDAS* system [55], and the most well known commercialized system is the *OnStar* system designed by General Motors. Such systems require quick analysis

of the underlying data in order to make diagnostic characterizations in real time. Effective event-detection algorithms are required in order to perform this task effectively.

The stock market often creates large volumes of data streams which need to be analyzed in real time in order to make quick decisions about actions to be taken. An example of such an approach is the MobiMine approach [56] which monitors the stock market with the use of a PDA. Such methods can be used for a wide variety of applications such as knowing human movement trends [24], social image search [77], animal trends [83] grocery bargain hunting [38], or more general methods for connecting with other entities in a given neighborhood [82].

4.4 Environmental and Weather Data

Many satellites and other scientific instruments collect environmental data such as cloud cover, wind speeds, humidity data and ocean currents. Such data can be used to make predictions about long- and short-term weather and climate changes. Such data can be especially massive if the number of parameters measured are very large. The challenge is to be able to combine these parameters in order to make timely and accurate predictions about weather driven events. This is another application of event detection techniques from massive streams of sensor data.

In particular, such methods have found numerous applications in prediction of long-term climate change [40, 58, 67]. For example, one can use various environmental parameters collected by sensors to predict changes in sea surface temperatures, indicators specific to global warming, or the onset of storms and hurricanes. A detailed discussion on the application of such methods for climate and weather prediction may be found in [40].

5. Conclusions and Research Directions

Data streams are a computational challenge to data mining problems because of the additional algorithmic constraints created by the large volume of data. In addition, the problem of temporal locality leads to a number of unique mining challenges in the data stream case. This chapter provides an overview to the generic issues in processing data streams, and the specific issues which arise with different mining algorithms.

While considerable research has already been performed in the data stream area, there are numerous research directions which remain to be explored. Most research in the stream area is focussed on the one pass constraint, and generally does not deal effectively with the issue of temporal locality. In the stream case, temporal locality remains an

extremely important issue since the data may evolve considerably over time. Other important topics which need to be explored are as follows:

- Streams are often collected by devices such as sensors in which the data is often noisy and error-driven. Therefore, a key challenge is to effectively clean the data. This may involve either imputing or modeling the underlying uncertain data. This can be challenge, since any modeling needs to be done in real time, as large volumes of the data stream arrive.

- A related area of research is in using the modeled data for data mining tasks. Since the underlying data is uncertain, the uncertainty should be used in order to improve the quality of the underlying results. Some recent research addresses the issue of clustering uncertain data streams [7].

- Many recent applications such as privacy-preserving data mining have not been studied effectively in the context of data streams. It is often a challenge to perform privacy-transformations of continuously arriving data, since newly arriving data may compromise the integrity of earlier data. The data stream domain provides a number of unique challenges in the context of the privacy problem.

References

[1] Aggarwal C., Xie Y., Yu P. (2011) On Dynamic Data-driven Selection of Sensor Streams, *ACM KDD Conference*.

[2] Aggarwal C., Bar-Noy A., Shamoun S. (2011) On Sensor Selection in Linked Information Networks, *DCOSS Conference*.

[3] Abadi D., Madden S., Lindner W. (2005) REED: robust, efficient filtering and online event detection in sensor networks, *VLDB Conference*.

[4] Aggarwal C. (2007) Data Streams: Models and Algorithms, *Springer*.

[5] Aggarwal C., Procopiuc C, Wolf J. Yu P., Park J.-S. (1999) Fast Algorithms for Projected Clustering. *ACM SIGMOD Conference*.

[6] Aggarwal C. (2006) On Biased Reservoir Sampling in the presence of Stream Evolution. *VLDB Conference*.

[7] Aggarwal C., Yu P. (2008) A Framework for Clustering Uncertain Data Streams. *ICDE Conference*.

[8] Aggarwal C. (2003) A Framework for Diagnosing Changes in Evolving Data Streams. *ACM SIGMOD Conference*.

[9] Aggarwal C. (2002) An Intuitive Framework for understanding Changes in Evolving Data Streams. *IEEE ICDE Conference*.

[10] Aggarwal C., Han J., Wang J., Yu P (2003). A Framework for Clustering Evolving Data Streams. *VLDB Conference*.

[11] Aggarwal C., Han J., Wang J., Yu P (2004). A Framework for High Dimensional Projected Clustering of Data Streams. *VLDB Conference*.

[12] Aggarwal C., Yu P. (2006) A Framework for Clustering Massive Text and Categorical Data Streams. *SIAM Data Mining Conference*.

[13] Aggarwal C, Han J., Wang J., Yu P. (2004). On-Demand Classification of Data Streams. *ACM KDD Conference*.

[14] Aggarwal C. (2009). *Managing and Mining Sensor Data*, Springer.

[15] Aggarwal C., Yu P. (2007). On Density-based Transforms for Uncertain Data Mining, *ICDE Conference*, 2007.

[16] Agrawal R., Imielinski T., Swami A. (1993) Mining Association Rules between Sets of items in Large Databases. *ACM SIGMOD Conference*.

[17] Alon N., Gibbons P., Matias Y., Szegedy M. (1999) Tracking Joins and Self-Joins in Limited Storage. *ACM PODS Conference*.

[18] Alon N., Matias Y., Szegedy M. (1996) The Space Complexity of Approximating Frequency Moments. *The Space Complexity of Approximating Frequency Moments*, pp. 20–29.

[19] Arici T., Akgun T., Altunbasak Y. (2006) A prediction error-based hypothesis testing method for sensor data acquisition. *ACM Transactions on Sensor Networks (TOSN)*, Vol. 2, pp. 529–556.

[20] Babcock B., Datar M., Motwani R. (2002) Sampling from a Moving Window over Streaming Data. *SIAM Symposium on Discrete Algorithms (SODA)*.

[21] Babcock B., Olston C. (2003) Distributed top-k monitoring, *ACM SIGMOD Conference*.

[22] Babu S., Widom J. (2001) Continuous queries over data streams. *SIGMOD Record*, 30(3), pp. 109–120.

[23] Bonnet P, Gehrke J., Seshadri P. (2001) Towards sensor database systems, *International Conference on Mobile Data Management*.

[24] Calabrese F., Kloeckl K., Ratti C. (2007) Wikicity: Real-Time Urban Environments. *IEEE Pervasive Computing*, 6(3), pp. 52-53. `http://senseable.mit.edu/wikicity/rome/`

[25] Cao F., Ester M., Qian W., Zhou W. (2006) Density-based Clustering of an Evolving Data Stream with Noise. *SIAM Data Mining Conference*.

[26] Chandrasekaran S., Franklin M. (2002) Streaming queries over streaming data. *VLDB Conference*.

[27] Chang J. H., Lee W. S. (2003) Finding recent frequent itemsets adaptively over online data streams. *ACM KDD Conference*.

[28] Chen Y., Tu L. (2007) Density-based Clustering for Real Time Stream Data, *ACM KDD Conference*.

[29] Chi Y., Wang H., Yu P., Muntz R. (2004) Moment: Maintaining closed frequent itemsets over a stream sliding window. *ICDM Conference*.

[30] Cormode G., Garofalakis M. (2005) Sketching Streams Through the Net: Distributed Approximate Query Tracking. *VLDB Conference*.

[31] Cormode G., Muthukrishnan S. (2004) An Improved Data Stream Summary: The Count-Min Sketch and its Applications. *LATIN*, pp. 29–38.

[32] Cormode, G., Muthukrishnan, S., Zhuang, W. (2007) Conquering the divide: Continuous clustering of distributed data streams. *ICDE Conference*.

[33] Datar M., Gionis A., Indyk P., Motwani R. (2002) Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813.

[34] Deligiannakis A., Kotidis Y., Roussopoulos N. (2004) Compressing Historical Information in Sensor Networks. *ACM SIGMOD Conference*.

[35] Deligiannakis A., Kotidis Y. (2005) Data Reduction Techniques in Sensor Networks. *IEEE Data Engineering Bulletin* 28(1): pp. 19–25.

[36] Deshpande A., Guestrin C., Madden S, Hellerstein J., Hong W. Model-driven data acquisition in sensor networks, *VLDB Conference*, 2004.

[37] Dasu T., Krishnan S., Venkatasubramaniam S., Yi K. (2005). An Information-Theoretic Approach to Detecting Changes in Multi-dimensional data Streams. *Duke University Technical Report CS-2005-06*.

[38] Deng L., Cox L. (2009) Livecompare: grocery bargain hunting through participatory sensing. *HotMobile*.

[39] Domingos P., Hulten G. (2000) Mining High-Speed Data Streams. In *Proceedings of the ACM KDD Conference*.

[40] Garg A. et al (2011) Gopher: Global observation of Planetary Health and Ecosystem Resources. *IGARSS*, pp. 1449–1452.

[41] Garofalakis M., Gehrke J., Rastogi R. (2002) Querying and mining data streams: you only get one look (a tutorial). *SIGMOD Conference*.

[42] Gianella C., Han J., Pei J., Yan X., Yu P. (2002) Mining Frequent Patterns in Data Streams at Multiple Time Granularities. *NSF Workshop on Next Generation data Mining*.

[43] Guha S., Mishra N., Motwani R., O'Callaghan L. (2000). Clustering Data Streams. *IEEE FOCS Conference*.

[44] Giannella C., Han J., Pei J., Yan X., and Yu P. (2002) Mining Frequent Patterns in Data Streams at Multiple Time Granularities. *Proceedings of the NSF Workshop on Next Generation Data Mining*.

[45] Golovin D., Faulkner M., Krause A. (2010) Online distributed sensor selection. *IPSN Conference*.

[46] Holter N., Generelli J. (1949) Remote recording of physiologic data by radio. *Rocky Mountain Medical Journal*, pp. 747-751.

[47] Hull B., Bychkovsky V., Chen K., Goraczko M., Miu A., Shih E., Zhang Y., Balakrishnan H., Madden S. (2006) CarTel: A Distributed Mobile Sensor Computing System, *ACM SenSys*.

[48] Hulten G., Spencer L., Domingos P. (2001) Mining Time Changing Data Streams. *ACM KDD Conference*.

[49] Indyk P. (2000) Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. *IEEE FOCS*.

[50] Jin R., Agrawal G. (2003) Efficient Decision Tree Construction on Streaming Data. *ACM KDD Conference*.

[51] Ioannidis Y., Poosala V. (1995) Balancing Histogram Optimality and Practicality for Query Set Size Estimation. *ACM SIGMOD Conference*.

[52] Jagadish H., Koudas N., Muthukrishnan S.,Poosala V.,Sevcik K., Suel T. (1998) Optimal Histograms with Quality Guarantees. *VLDB Conference*.

[53] Jin R., Agrawal G. (2005) An algorithm for in-core frequent itemset mining on streaming data. *ICDM Conference*.

[54] Johnson W., Lindenstrauss J. (1984) Extensions of Lipshitz mapping onto Hilbert Space. *Contemporary Mathematics*, Vol 26, pp. 189–206.

[55] Kargupta H. et al VEDAS: A Mobile and Distributed Data Stream Mining System for Vehicle Monitoring. *SDM Conference*, 2004.

[56] Kargupta H. et al MobiMine: Monitoring the stock market using a PDA, *ACM SIGKDD Explorations*, January 2002.

[57] Kasetty S., Stafford C., Walker G., Wang X., Keogh E. (2008) Real-Time Classification of Streaming Sensor Data, *ICTAI Conference*.

[58] Kawale J., Steinbach M., Kumar V. (2011) Discovering Dynamic Dipoles in Climate Data. *SDM Conference*.

[59] Kifer D., David S.-B., Gehrke J. (2004). Detecting Change in Data Streams. *VLDB Conference*, 2004.

[60] Kollios G., Byers J., Considine J., Hadjieleftheriou M., Li F. (2005) Robust Aggregation in Sensor Networks. *IEEE Data Engineering Bulletin*.

[61] Krause A., Guestrin C. (2007) Near-optimal observation selection using submodular functions. *AAAI Conference*.

[62] Krause A., Horvitz E., Kansal A., Zhao F. (2008) Toward Community Sensing. *IPSN*, pp. 481–492.

[63] Madden S., Franklin M., Hellerstein J. (2005) TinyDB: an acquisitional query processing system for sensor networks, *ACM Transactions on Database Systems*, 30(1), pp. 122–173.

[64] Manku G., Motwani R. (2002) Approximate Frequency Counts over Data Streams. *VLDB Conference*.

[65] Medioni G., Cohen I., Bremond F., Hogeng S., Nevatia R. (2001) Event Detection and Analysis from Video Streams, *IEEE TPAMI*, 23(8).

[66] Olston C., Jiang J., Widom J. (2003) Adaptive Filters for Continuous Queries over Distributed Data Streams. *SIGMOD Conference*.

[67] Race C., Steinbach M., Ganguly A., Semazzi F., Kumar V. (2010) A Knowledge Discovery Strategy for Relating Sea Surface Temperatures to Frequencies of Tropical Storms and Generating Predictions of Hurricanes Under 21st century Global Warming Scenarios, *CIDU*, pp. 204–212.

[68] Rodrigues P., Gama J., Lopes L. (2008) Clustering Distributed Sensor Data Streams, *PKDD Conference*.

[69] Sakaki T., Okazaki M., Matsuo Y. (2010) Earthquake shakes Twitter users: real-time event detection by social sensors, *WWW Conference*.

[70] Sakurai Y., Papadimitriou S., Faloutsos C. (2005). BRAID: Stream mining through group lag correlations. *ACM SIGMOD Conference*.

[71] Sung M., Marci C., Pentland A. (2005) Wearable Feedback Systems for Rehabilitation, *Journal of Neuroengineering and Rehabilitation*, 2:17.

[72] Vitter J. S. (1985) Random Sampling with a Reservoir. *ACM Transactions on Mathematical Software*, Vol 11(1), pp. 37–57.

[73] Waltz E., Llinas J. (1990) Multi-Sensor Data Fusion, *Artech House Radar Library*.

[74] Wang H., Fan W., Yu P., Han J. (2003) Mining Concept-Drifting Data Streams using Ensemble Classifiers. *ACM KDD Conference*.

[75] Yi B.-K., Sidiropoulos N.D., Johnson T., Jagadish, H. V., Faloutsos C., Biliris A. (2000). Online data mining for co-evolving time sequences. *ICDE Conference*.

[76] Xue W., Luo Q., Chen L., Liu Y. (2006) Contour Map Matching for Event Detection in Sensor Networks, *ACM SIGMOD Conference*.

[77] Yan T., Kumar V., Ganesan D. (2010) Crowdsearch: Exploiting crowds for accurate real-time image search on mobile phones, *MobiSys*.

[78] Yao Y., Gehrke J. (2003) Query processing in sensors networks, *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*.

[79] Yin J., Yang Q., Pan J. (2008) Sensor-based Abnormal Human Activity Detection, *IEEE TKDE*, 20(8), pp 1082–1090.

[80] Yu J. X., Chong Z., Lu H., Zhou A. (2004) False positive or false negative: Mining frequent itemsets from high speed transaction data streams. *VLDB Conference*.

[81] Zhang T., Ramakrishnan R., Livny M. (1996) BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD Conference*.

[82] `http;//www.wikitude.com`

[83] `http://www.movebank.org`

Chapter 7

REAL-TIME DATA ANALYTICS IN SENSOR NETWORKS

Themis Palpanas
University of Trento
Trento, Italy
themis@disi.unitn.eu

Abstract The proliferation of Wireless Sensor Networks (WSNs) in the past decade has provided the bridge between the physical and digital worlds, enabling the monitoring and study of physical phenomena at a granularity and level of detail that was never before possible. In this study, we review the efforts of the research community with respect to two important problems in the context of WSNs: real-time collection of the sensed data, and real-time processing of these data series.

Keywords: Wireless Sensor Networks, Data Collection, Data Analytics, Outliers, Deviation Detection, Uncertain Data Series, Data-Aware Network Protocols

1. Introduction

In the past decade, we have witnessed the proliferation of Wireless Sensor Networks (WSNs), fueled by advances in processor technologies and wireless communications that led to the development of small, low cost and power efficient sensor nodes [100, 50, 74]. The great benefit they provide is that they serve as the bridge between the physical and digital worlds, and they enable us to monitor and study physical phenomena at a granularity and level of detail that was never before possible.

Collecting the data sensed by the WSN to a centralized server (the sink), or being able to directly query the WSN are probably the most important functionalities that a WSN has to support. Lots of work has been directed to how to efficiently achieve these goals, where the primary

objective is to extend the WSN lifetime, while fulfilling the application requirements (collecting the required data, or answering the queries).

There are two main ideas that researchers have explored: first, data are correlated (both across time and over space), and second, several applications accept small errors in the data values they operate on. These ideas have led to the development of a multitude of techniques that trade accuracy for time performance and energy savings.

In this study, we review the efforts of the research community with respect to the problems of real-time collection of the sensed data, and real-time processing of these data series in the context of a WSN. Furthermore, we examine the interplay between such data management techniques and network protocols.

We note that the aim of this study is not an exhaustive enumeration and discussion of all the related works, but rather, the description of prominent research problems that have been studied so far with regards to the sensor data processing and analysis, as well as of promising future research directions.

2. Data Collection

The availability and use of sensor networks have generated a lot of research interest. A major part of this effort has concentrated on how to collect the sensed data at the sink (where they will be further processed and analyzed), using the least amount of energy[1] possible. The challenge arises from the special characteristics of WSNs and the nature of the data they produce, namely: limited resources, intermittent connections, and spatio-temporal correlation of the sensed values [60, 56, 101].

Several frameworks for the efficient execution of queries and collection of data in a sensor network have been developed in the last years [60, 59, 103]. The focus in these works was to propose data processing and optimization methods geared specifically toward sensor networks (we describe those in detail later on). The early studies described in-network aggregation techniques for reducing the amount of data transmitted by the nodes, while subsequent research focused on model-driven [32] and data-driven [87] data acquisition techniques. Other works have proposed techniques that take into account missing values, outliers, and intermittent connections [44, 30, 101, 88].

A different approach is based on Kalman filters [51], with the same goal of reducing the required communication among nodes and the sink.

[1] Given that radio communication in WSNs is much more expensive than CPU processing, this translates to reducing communication and data transfer.

Other techniques offer solutions for efficient spatio-temporal data suppression [56, 113, 99, 52, 47, 73], where in addition to the temporal correlations present in the sensor network data, they aim at identifying and exploiting the spatial correlations of the data, as well. Furthermore, previous works have proposed algorithms that help in the selection of representative nodes when we want to monitor large-scale phenomena (i.e., phenomena that evolve over days, or months, and involve several sensor nodes) [6], or when we want to take into account the remaining energy of each individual node [63]. The above techniques help to further reduce the communication cost of the sensor network, and could be applied on top of the model-driven, or data-driven techniques.

In the rest of this section, we will discuss techniques in the areas of model-driven and data-driven data acquisition, as well as in spatio-temporal data suppression.

2.1 Model-Driven Data Acquisition

The aim of the model-driven approach is to (conceptually) collect, or process queries on all the data sensed by the WSN, based on probabilistic models that capture the correlations that exist in these data. We note that sensor readings exhibit such correlations in a wide range of domains and applications. This is true, because often times sensors are monitoring slow-changing phenomena with high temporal resolution and/or high spatial resolution. Moreover, correlations may also be present among different types of readings coming from the same sensor node (e.g., it has been shown that temperature and voltage readings are correlated [32]; at the same time it is much less expensive to take voltage readings than temperature).

The model-driven approach works as follows. During an initial training phase, all the sensed data are collected from the nodes in the network, in order to train the probabilistic models that are stored in the sink. Then, these models are used in order to estimate the sensed values, and additionally provide probabilistic guarantees on the correctness of these estimates. Therefore, instead of querying the sensors, we operate on the data produced by the models. If the guarantees produced by the models for these data do not satisfy the accuracy requirements of the application, then we can request additional real data values from the sensors, in order to refine the models to the point that the probabilistic guarantees satisfy the application requirements.

We can now formally define the model-driven data acquisition problem.

PROBLEM 2.1 (MODEL-DRIVEN DATA ACQUISITION) *Given a sensor network, and a sink that needs to collect all the sensed values within ϵ of the real value with confidence (probability) at least $1 - \delta$, design a data collection protocol such that the energy used by the sensor network is minimized.*

In order to solve this problem, we need to decide on the probabilistic models to use for approximating the distributions of the sensed values, and also on the communication strategies among the sensors and the sink. Both these aspects of the problem are addressed by the studies that we discuss in the next paragraphs.

2.1.1 Proposed Techniques. The BBQ system [32] proposes sensor data acquisition techniques based on time-varying multivariate Gaussian probabilistic models, but other models can alternatively be used, such as probabilistic graphical models [31]. Using the above approach, the produced models capture correlations both among sensed values from the same sensor across time, and among different sensors across space. We note that the above approach requires some knowledge of the special characteristics of the data distribution, such as periodic drifts, which should be encoded in the space of models considered. This means that some minimum amount of domain knowledge is required, in order to make effective use of these techniques.

A similar framework for modeling sensor network data is proposed by Guestrin et al. [45]. The goal is for groups of nodes in the network to collaborate in order to fit a global function to each of their local measurements. This approach employs kernel linear regression in order to model the sensed values, by capturing spatio-temporal correlations. Once again, we observe that this is a parametric approximation technique, and as such, requires the user to make an assumption about the number of estimators required to fit the data. Moreover, there is a need for a training phase (where the models are built, evaluated, and adjusted), which in practice can be rather lengthy and expensive.

Even though the domain knowledge requirement that the above techniques have may be prohibitive for some applications, we note that a large number of applications (where the measured phenomena are known or understood, or when a domain expert is available) can still benefit from such techniques.

2.2 Data-Driven Data Acquisition

The model-driven approach described earlier can lead to significant energy savings for the data acquisition task. However, by the nature

of their techniques, they can only provide probabilistic guarantees on the accuracy of the data that the sink collects, and hence no absolute bound on the error. While this may be sufficient for certain applications (e.g., temperature and humidity monitoring for Heating, Ventilation and Air Conditioning systems), there exists a class of applications, for which hard accuracy guarantees are essential (e.g., scientific applications that need accurate, fine-grained monitoring of some phenomenon).

In several scientific applications, it may also be the case that the domain experts do not already have a model of the data distribution they are sampling using the WSN, but are rather interested in collecting accurate measurements in order to build such a model [19]. Indeed, WSNs offer a unique opportunity to scientists to observe phenomena and develop models for them at a scale and granularity that were never before possible. Nevertheless, in order to so, they need to have accuracy guarantees on the sensor measurements.

In data-driven data acquisition, we make the assumption that the application running at the sink allows for a small tolerance in the accuracy of the reported data. In contrast with the ideal requirements of the sink obtaining *exact* values in *all* data reports, the correctness of these applications is unaffected as long as *i)* the reported values match *closely* the exact ones; *ii)* inaccurate values occur only *occasionally*. In other words, deviations from the exact reports are acceptable, as long as their extent in terms of difference in *value* and *time interval* during which the deviation occurs are small enough. We capture these assumptions, common to many applications, with the following definitions on value tolerance, ε_V, and time tolerance, ε_T (refer to Figure 7.1). We use the term *error tolerance*, ε_{VT} to refer to both of them together.

Definition 7.1 (Value Tolerance) *Let V_i be an exact measurement taken at time t_i. The* value tolerance *is defined by the maximum relative and absolute errors acceptable, $\varepsilon_V = (\epsilon^{rel}, \epsilon^{abs})$. From the application perspective, reading a value V_i becomes equivalent to reading any value $\hat{V}_i$ in the range R_V defined by the maximum error, $\hat{V}_i \in R_V = [V_i - \epsilon, V_i + \epsilon]$, where $\epsilon = \max\{\frac{V_i}{100}\epsilon^{rel}, \epsilon^{abs}\}$. In other words, the application considers a value $\hat{V}_i \in R_V$ as* correct.

Note that the value tolerance includes both an absolute and a relative component. This is useful for applications that involve sensor readings with wide ranges of values.

Definition 7.2 (Time Tolerance) *Let $T = |t_j - t_k|$ be a time interval, and $\hat{V}_T = \{\hat{V}_j, \ldots, \hat{V}_k\}$ the set of values reported to the application during T. The* time tolerance *ε_T is the maximum acceptable value of*

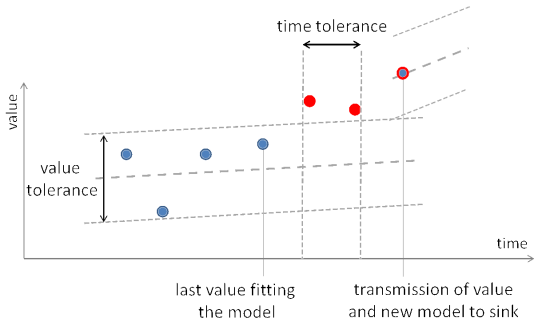


Figure 7.1. Value and time tolerance, assuming a linear model (depicted by the thick dashed line) for the sensed data [79].

T such that all *the values reported in this interval are incorrect, i.e.,* $\hat{V}_i \notin R_V$, $\forall \hat{V}_i \in \hat{V}_T$.

Similarly to the model-driven approach, each node (or group of nodes) in the WSN generates a model for the sensed data. This model is then sent to the sink, along with the last reading. From that point on, the sink can predict the readings of the node based on this shared model. The node is also checking whether its model can accurately describe its own readings (within the error tolerance agreed with the sink), and if this is not true then it computes a new model and transmits it to the sink. Evidently, the sink always records accurate data (i.e., within ε_{VT}), regardless of the quality of the model. The model quality affects only the effectiveness of the proposed scheme in terms of energy savings.

We can now formally define the problem of data-driven data acquisition.

PROBLEM 2.2 (DATA-DRIVEN DATA ACQUISITION) *Given a sensor network, and a sink that needs to collect all the sensed values within* ε_{VT}*, design a data collection protocol such that the energy used by the sensor network is minimized.*

This problem statement is deliberately vague on the specificities of the design of such a protocol. In the following paragraphs we review several techniques that solve this problem, each one focusing on different aspects of the problem. Some studies focus on the selection of the sensed data model (shared among sensors and sink), others concentrate on the effective identification of temporal and/or spatial correlations among the

sensed data, while others explicitly aim at maximizing the lifetime of the *entire* sensor network[2].

2.2.1 Proposed Techniques. The KEN technique [25] builds and maintains dynamic probabilistic models over the sensor readings, taking into account the spatio-temporal correlations that exist in the sensor readings. These models organize the sensor nodes in non-overlapping groups, and are shared by the sensor nodes and the sink. The expected values of the probabilistic models are the values that are recorded by the sink. If the sensors observe that these values are more than ε_{VT} away from the sensed values, then a model update is triggered.

The PAQ [98] and SAF [97] methods employ linear regression and autoregressive models, respectively, for modeling the measurements produced by the nodes, with SAF leading to a more accurate model than PAQ.

Silberstein et al. [86, 87] describe for providing continuous data without continuous reporting, but with checks against the actual data. To achieve this goal, this approach introduces temporal and spatio-temporal suppression schemes, which use the in-network monitoring to reduce the communication rate to the central server. Based on these schemes, data is routed over a chain architecture. At the end of this chain, the nodes that are most near to central server send the aggregate change of the data to it. Since in this scheme (and in data-driven approaches in general) the loss of a model update is crucial[3], special provision is taken for handling network failures [87], so as to ensure correctness.

A recent study proposes a new linear model, DBP [79]. The model is trained using m data points, where the first and the last l points are called *edge points*, and is computed as the slope δ of the segment that connects the average values over the l edge points at the beginning and end of the training phase. This model mitigates the problem of noise and outliers: instead of trying to reduce the approximation error to the *data points* in the recent past, DBP aims at producing models that are consistent with the *trends* in the recently-observed data. Consequently, it leads to improved performance, especially in noisy settings. Moreover, the computation of this model is very simple, and therefore appealing for implementation on resource-scarce nodes.

[2]Note that by minimizing the energy consumption of the network, it is possible that the energy of a few specific sensor nodes is depleted much faster than the average. Obviously, this is not desirable, since it may jeopardize the correct operation of the entire network.

[3]Losing a single model-update message has the potential to introduce large errors at the sink, as the latter will continue to predict sensor values with an out-of-date model until the next one is received.

Another idea that has been studied is to select a set of representative nodes, and use only those for transmitting measurements to the sink. The premise is that each representative node has measurements similar to the measurements of the nodes in its neighborhood. Then, it is only the representative nodes that need to communicate the sensed values to the sink, thus, significantly reducing the energy spent by the WSN.

Data mining approaches contributed to this problem, by providing techniques for clustering and selecting representatives [46, 80, 62, 108]. Inside each cluster, the node with the most similar readings to the measurements of all nodes inside that cluster is selected as a cluster representative. Many algorithms were developed to deal with the online distributed clustering of data.

SERENE [9] is a framework for SElecting REpresentatives in a sensor NEtwork. It uses clustering techniques to select the subset of nodes that can best represent the rest of sensors in the network. In order to select an appropriate set of representative sensors, SERENE performs an analysis of the historical readings of sensor nodes, identifies the spatio-temporal correlations among sensors (based on their readings), and groups sensors into clusters according to these correlations. Then, each cluster performs further analysis in order to select the sensors with the highest representation quality. We note that the analysis of the historical data, which has to be repeated when the distribution of the sensor readings changes, may take place in the sensors or in the sink, according to the amount of resources required.

Snapshot Queries [56] is another approach that introduces a platform for energy efficient data collection in sensor networks. By selecting a small set of representative nodes, this approach provides responses to user queries and reduces the energy consumption in the network. In order to select representatives, each sensor node in this approach builds a data model of the distribution of measurement values of its neighbors for each attribute. After a node decides which of its neighbors it can effectively represent, it broadcasts its list of candidate cluster members to all its neighbors. Each node selects as its representative the neighbor that can represent it, and that additionally has the longest list of candidate cluster members.

In ECLUN [47], nodes do not continuously communicate with the representatives, but communication is established only when a state change is detected in the monitored phenomena. This communication is further reduced through the careful construction of clusters, which considers similarity in sub-spaces of the full-dimensional sensor readings space. This makes the above approach suitable to deployments of sensor node that produce multi-dimensional readings (i.e., monitor several phenom-

ena simultaneously). ECLUN also tries to uniformly distribute the energy usage among the nodes, resulting in a longer lifetime for the entire sensor network, since the variance of the lifetime of individual nodes is minimized.

A more recent study [4] focuses on the problem of identifying functional dependencies among sensor data streams, in order to determine a small number of sensors from which data are actively collected. The rest of the sensors collect data at lower rates, with the purpose of detecting changes in the discovered dependencies and taking actions to reorganize the sensor data collection process. The dependencies identified in this work are based on regression analysis that takes into account possible lags among the streams.

The above studies use different ways of calculating the correlation among the sensor streams in the network. For this part of the problem, other techniques for identifying correlations in multiple data streams [107, 114, 72, 26, 82] could be used as well. The work by Aggarwal et al. [3] describes a method that additionally considers and exploits domain-specific knowledge on the information network of the sensors (i.e., relating to links among the sensors). Another approach for the same problem has proposed a technique for selecting sensors that is based on feedback on the utility of the selected sensors [43].

2.3 Data Series Summarization

Many sensor network applications in diverse domains produce voluminous amounts of data series, such as in meteorology (e.g., temperature measurements [1]), oceanography (e.g., water level measurements [90]), and other domains. The sheer number and size of the data series we need to manipulate in many of the real-world applications mentioned above dictates in several cases the need for a more compact representation of data series than the raw data itself, and a plethora of representations have been proposed to that effect[4].

Even though most data series representations treat every point of the data series equally, there exist WSN applications for which the time position of a point makes a difference in the fidelity of its approximation. Then we would represent the most recent data with low error, and would

[4]Several techniques have been proposed in the literature for the approximation of data series, including *Discrete Fourier Transform (DFT)* [76, 36], *Discrete Cosine Transform (DCT)*, *Piecewise Aggregate Approximation (PAA)* [106], *Discrete Wavelet Transform (DWT)* [75, 21], *Adaptive Piecewise Constant Approximation (APCA)* [20, 58], *Piecewise Linear Approximation (PLA)* [54], *Piecewise Quadratic Approximation (PQA)* [48], and others. Most of them are amenable to incremental, online operation.

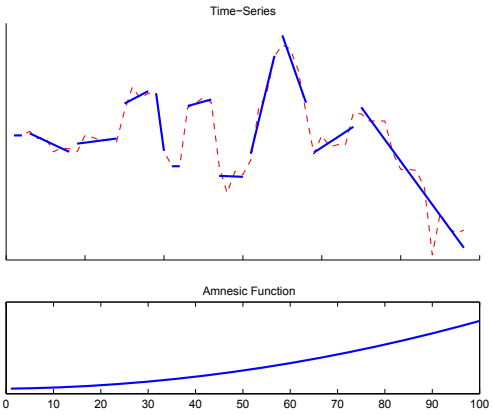


Figure 7.2. Depiction of an amnesic approximation, using the piecewise linear approximation technique (the most recent values of the data series are on the left; the oldest values are on the right) [70].

be more forgiving of error in older data. We call this kind of time series approximation *amnesic*, since the fidelity of approximation decreases with time, and it therefore requires less memory for the events further in the past (see Figure 7.2).

For example, the Environmental Observation and Forecasting System[5] [90] operates in a way that allows for some sensors only intermittent connections to the sink (through a repeater station that is not always available). Since the station does not know how long it will be offline, and has a finite buffer, amnesic approximation is an effective way to record the data.

We need a way to specify for each point in time the amount of error allowed for the approximation of the time series. In order to achieve this goal, we use the *amnesic function* $A(x)$, which returns the acceptable approximation error for every point of the data series. We define two forms of amnesic functions, namely, the *relative* and the *absolute* amnesic functions. A relative amnesic function determines the relative approximation error we can tolerate for every point in the time series (e.g., we can specify that when we approximate a point that is twice as old, we will accept twice as much error). When we use relative amnesic functions, we fix the amount of memory that we are allowed to use for the approximation of the data. On the other hand, an absolute am-

[5]This is a large-scale distributed system designed to monitor, model, and forecast wide-area physical processes, such as river systems.

nesic function specifies, for every point in the data series, the *maximum* allowable error for the approximation, which is useful when the application requires quality guarantees for the approximation of the data series. When we use absolute amnesic functions, we allow the approximation to use as much memory as necessary in order to meet the error bounds.

More formally, we define the following two problems in the context of *landmark windows*. The *landmark window* is the window that contains all the values of the data series (from a given time point) up to now.

Problem 2.3 (Land. Win. with Relative Amnesic (URA))
Given a memory budget M and a relative amnesic function $RA(x)$, construct an amnesic approximation using memory at most M that minimizes the approximation error of the data points inside the window.

Problem 2.4 (Land. Win. with Absolute Amnesic (UAA))
Given an absolute amnesic function $AA(x)$, construct an amnesic approximation that minimizes the required memory M.

Note that in the *URA* and *UAA* problems, the optimization objective is different. In the *URA* problem we seek to minimize the approximation error given the memory space used by the data series approximation, while in the *UAA* problem we want to minimize the space used in the approximation given the maximum error allowed.

Following the definition of the problems for the landmark window, we now define the corresponding problems for the case where we consider the sliding window model.

Problem 2.5 (Sliding window with Relative Amnesic (SRA))
Given a sliding window W, a memory budget M, and a relative amnesic function $RA(x)$, construct an amnesic approximation using memory M that minimizes the approximation error of the data series within the sliding window.

Problem 2.6 (Sliding window with Absolute Amnesic (SAA))
Given a sliding window W, and an absolute amnesic function $AA(x)$, construct an amnesic approximation that minimizes the required memory M.

2.3.1 Proposed Techniques. Bulut and Singh propose the use of wavelets to represent data streams, which are biased towards the more recent values [16], and describe an efficient, online method for incrementally maintaining this representation. The bias to the most recent values can be seen as a special case of an amnesic function, whose

form in this particular case is dictated by the hierarchical nature of the wavelet transform.

A subsequent study [111] generalizes on these ideas, by decoupling the approximation of the time series from a particular dimension-reduction algorithm, and employs user-input to specify how the available memory will be used for the approximation. There has also been relevant work in machine learning, and more specifically, in the neural network community, where the main goal is to model time-varying patterns in data series [10, 29].

A general and efficient solution to the amnesic summarization problems defined earlier is presented in [70]. This study describes solutions for the four variations of the problem, based on online algorithms that use a piecewise linear approximation model. When a new point arrives, the algorithms update the approximation model in sub-linear time on the number of linear segments.

It has been shown that the techniques mentioned above can be implemented in a very efficient manner in sensor nodes [89]. Moreover, amnesic summarization has been studied in the context of flash memories [67], which offer significant benefits that can be exploited by WSN deployments.

3. Data Processing

Another interesting and important research direction in the context of WSN data management is that of efficient data processing and analysis, and a significant amount of effort has been devoted to it. In this case, we are interested in supporting different types of complex queries in the specific, resource-constrained environment of a WSN.

Several frameworks for the efficient execution of queries in a sensor network have been developed in the past years [60, 59, 103]. The focus in these works was to propose data processing and optimization methods geared specifically towards sensor networks, with the early studies describing in-network aggregation techniques for reducing the amount of data transmitted by the nodes. Ali et al. [7] propose an interesting approach to detect and track discrete phenomena (PDT) in sensor networks. Hellerstein et al. [49] propose algorithms to partition the sensors into *isobars*, i.e., groups of neighboring sensors with approximately equal values during an epoch. Other works have proposed techniques that take into account missing values, outliers, and intermittent connections [44, 30, 101]. We note that some of the techniques we discussed earlier are applicable here (e.g., either to answer adhoc queries [31], or `SELECT`* queries [87]).

In the following paragraphs, we present a framework that enables the development of a variety of complex processing applications in a sensor network. These are applications with high processing requirements over a significant portion of the data generated by the entire WSN. Examples of such applications are the identification and tracking of homogeneous regions, and outlier detection. The identification and tracking of homogeneous regions is used for environmental monitoring (e.g., around oil-drill, or chemical plant sites). In outlier detection, we are interested in discovering exceptional situations that may require the attention of a human analyst: when some of the values of some sensor are not normal, when the number of abnormal values exceeds a given threshold, or when the values of a given sensor are significantly different from the values of its neighbors. We further discuss these applications below.

3.1 Enabling Complex Analytics

The way that streaming applications are able to efficiently process continuous data arriving at high rates, such as those generated by WSNs, is by computing succinct summaries of the data, and operating on these summaries [41, 32].

The framework we describe below aims to approximate in an online fashion multi-dimensional data series distributions [69]. This framework is adaptive and does not require any a priori knowledge about the distributions of the sensed values. Moreover, it operates in a distributed fashion, thus, exploiting all the available resources of the WSN, and reusing any processing that has already taken place.

3.1.1 Data Distribution Approximation Framework.

The proposed framework for estimating the underlying distribution of a streaming data series works both for the sliding time window and the landmark window models [69]. This framework estimates the distribution of the values generated by the sensors using the kernel density estimators [84], which offer the following desirable properties: (i) they are efficient to compute and maintain in a streaming environment; (ii) they can very accurately approximate an unknown data distribution, with no a priori knowledge and (effectively) no parameters; (iii) they can easily be combined and (iv) they scale well in multiple dimensions. The above properties make the framework applicable to large sensor networks, organized in a hierarchical way[6] [104].

[6]The hierarchical decomposition of the sensor network, as well as the selection of the leaders for each level of the hierarchy, can be achieved using any of the energy-efficient techniques proposed in the literature [38, 61, 110].

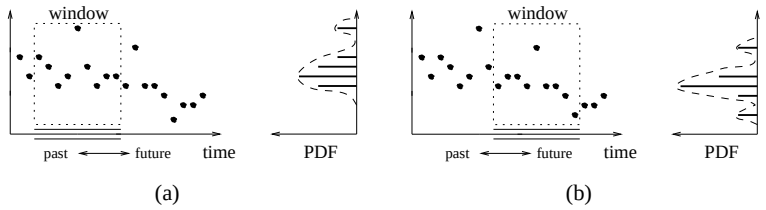


Figure 7.3. Estimation of data distribution in sliding window for two time instances (1-d data) [69].

In such an online setting, we require that each sensor maintains a model for the distribution of values it generates within a sliding window W (see Figure 7.3). Such a model can be efficiently and effectively maintained over time. Then, we need to ensure that this mechanism operates in a distributed fashion. Through a model composition mechanism, we are able to take the data distribution models of two (or more) streams, and construct a single model that describes their combined behavior. The framework also proposes mechanisms for incrementally maintaining the models across all levels of the (conceptual) hierarchy, as well as for comparing them in order to determine the similarity of the sensed values. All the above operations can be efficiently supported in real-time by a sensor node [69].

3.2 Detection and Tracking of Homogeneous Regions

The first application is identification and tracking of *homogeneous regions* [7, 49], which are defined as spatial divisions of the field under observation that exhibit similar measured values over time, such as an oil spill detected in the ocean (see Figure 7.4). The sensors deployed around the origin of the spill can organize themselves into a network and communicate the measurements, to detect regions of varying oil concentrations.

Recent studies propose methods for delineating homogeneous regions by a boundary [24, 68]. However, in several situations we need a more generalized grouping of the sensors, based on the sensed values over a time interval. In general, we would like to solve the problems of detecting and tracking such homogeneous regions in *real-time* when the definition of the phenomenon is *not* known in advance.

Using the framework described in Section 3.1.1, we can efficiently identify sensors with similar readings, by comparing their models of the densities of the sensed values [92]. Sensors with very similar models (i.e.,

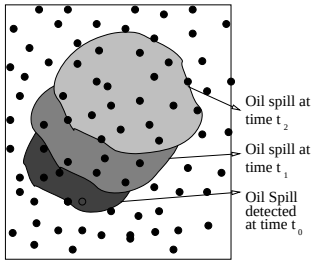


Figure 7.4. Spread of an oil spill detected in the water over time [92].

data distributions) are grouped together, using the hierarchical organization of the WSN. Each group corresponds to a homogeneous region in space, whose boundaries can be effectively approximated. Then, we can track the movements of these regions over time in a distributed manner, keeping awake only the sensors that are close to the regions that are being tracked. This process is efficiently implemented by tracking the movement of the boundaries of each region.

3.3 Outlier Detection

The second application, which we examine in more detail, is distributed deviation detection in a sensor network. The goal is to identify values (or the corresponding sensor nodes) that look very different from their spatio-temporal neighbors (i.e., the values in the recent history of the sensor stream, or the values in the streams of spatially close sensors). We note that this is a challenging problem, even for static datasets.

This problem is important in a WSN setting because it can be used to identify faulty sensors, and to filter spurious reports from different sensors. Even if we are certain of the quality of measurements reported by the sensors, the identification of outliers provides an efficient way to focus on the interesting events in the sensor network.

In the following subsections, we describe the approaches that have been proposed in the literature, separating them in *approximate* and *exact*, according to whether they provide guarantees on the detection of all the outliers.

3.3.1 Approximate Approaches. We first examine outlier identification techniques that cannot provide any hard guarantees on the

correctness of the results they produce. Consequently, these techniques may fail to report some of the outliers in the data.

Classification-based

A method based on Bayesian classifiers is described by Elnahrawy et al. [34]. This is a method for modeling and learning statistical contextual information in WSNs, which can also be applied for the task of outlier identification. The employed model assumes that the current reading of each sensor is only influenced by the preceding reading of the same sensor, and the readings of its immediate neighbors. This model is then used to predict the highest probability class of the subsequent reading. If the probability of this class is significantly different from the probability (according to the model) of the actual reading, then this reading is deemed an outlier.

Rajasegarar et al. [77] propose an alternative approach that uses a Support Vector Machine (SVM) classifier. In this case the classification model uses only the information from the past readings of the same sensor node, and ignores the readings from the neighboring nodes.

A drawback of the classification-based approaches is the time and computational effort required in order to train the model that can then be used for outlier detection. This effort can in certain cases be rather high. Note also that for non-stationary data this effort will be continuous.

Data Distribution-based

A technique for outlier detection, based on learning statistical properties of the spatio-temporal correlations of the sensor readings, is proposed by Bettencourt et al. [12]. This technique is geared towards ecological applications, where the sensed pheonomena evolve slowly over time, and are spatio-temporally coherent. According to this technique, sensors learn the distributions of differences among their own readings (over time), as well as the distributions of differences between their readings and the readings of their neighbors. Then, comparing the current readings to these distributions, allows sensors to identify local outliers using a significance test, and a user-specified threshold.

Subramaniam et al. [93] study the case where we wish to identify (among all sensor readings in a sliding window) those values that have very few near neighbors [55], namely, *distance-based* outliers; or those values whose near neighborhood is significantly less dense than their extended neighborhood [71], namely, *density-based* outliers. Note that these definitions do not require any prior knowledge of the underlying data distributions. In order to solve the problem (for both definitions

of outliers mentioned above), we need to count the number of sensed values that fall in different regions of the data space. This operation can be efficiently supported by the framework outlined in Section 3.1.1, and the overall task can be distributed in the entire WSN. Especially for the distance-based outliers, the following observation holds [93]. In a (conceptual) hierarchical organization of the sensor network, a parent node combines in a single pool all the data that its children process. Consequently, outliers have to be identified with respect to this new pool of data. Nevertheless, it is not necessary that the parent node reads in all the data from its children's input data streams, and for each data value determine whether it is an outlier or not. It suffices for the parent node to examine only the values that have been marked as outliers by its children. All the other data values can be safely ignored, since they cannot possibly be outliers. The above approach allows for the effective distribution of the outlier detection task to the entire WSN, resulting in significant savings in terms of communication messages.

A recent study [64] proposes the use of the hyperellipsoidal model in order to model the normal behavior of sensor nodes. Sensor readings that significant deviate from this model are then declared outliers. The focus of this study is on devising an iterative approach for building and maintaining hyperellipsoidal models, which makes them suitable for non-stationary data distributions.

Node Similarity-based

Zhuang et al. [115] describe an approach for identifying (and cleaning) outliers in a sensor network. They focus on two kinds of outliers: *short simple outliers*, usually represented as an abnormal, sudden burst and depression; and *long segmental outliers*, which represents erroneous sensor readings that last for a certain time period. Their approach works as follows. The Discrete Wavelet Transform (DWT) is applied on the series of sensor readings. The high-frequency coefficients are omitted from the resulting DWT representation, which is subsequently compared to the original data series. Data points that are further away than a distance threshold, d_1, from their DWT representation are deemed short outliers. Then, the data series is compared to the series obtained from other sensors that are geographically close. If no other series is within some distance threshold, d_2, then this data series is deemed a long outlier (similarity between data series is measured using the dynamic time warping distance [11]).

A similar problem is addressed by a subsequent study [102], which targets the identification of outlying sensors. The main observation is that sensors observing the same phenomenon are spatially correlated,

but outlying sensor readings are geographically independent. The algorithm described in this study has each sensor compute the difference of its reading to the median reading of its neighboring sensors. Then the sensor collects all these differences from its neighborhood and standardizes them. If the absolute value of its standardized difference is larger than a threshold, d, then this sensor is deemed an outlier.

The TACO framework [40] was recently proposed by Giatrakos et al. to operate in a WSN. In order to identify outliers, TACO takes into account both the history of measurements of a given sensor, as well as the spatial correlations with measurements of other sensors in the vicinity. The outlier detection scheme is based on a two-level hashing mechanism. The first level of hashing takes place locally in each sensor, and is based on Locality Sensitive Hashing [23]. This is used for dimensionality reduction, since the recent history of sensor data readings can be succinctly represented in a space of much smaller dimensionality. Assuming a clustered organization of the sensor network (i.e., hierarchical organization with just two levels), each node communicates this reduced representation of its history to the corresponding cluster-head, which subsequently checks for similar representations among the other nodes in the cluster. Similarity measures such as cosine similarity, Jaccard coefficient and correlation coefficient, are supported. The representations that do not find any similar matches make part of a list of potential outliers that is further communicated to all the cluster-heads of the sensor network. This communication step is efficiently implemented using a second hashing mechanism based on the hamming weight of the representations. Overall, the approach has the advantage that it can provide probabilistic guarantees on the accuracy of the results.

Giatrakos et al. [39] proposed a similar technique, only based on the trends of the sensed data series.

3.3.2 Exact Approaches. Unlike the works above, some studies have proposed techniques for outlier detection that guarantee no false negatives (i.e., they identify all outliers). This is a desirable property for several critical applications (e.g., structural integrity monitoring).

The work by Branch et al. [13] describes a technique for distributed outlier detection, where the goal is to identify global outliers (i.e., with respect to the data collected by all sensors). This technique supports definitions of outliers that conform to certain anti-monotonicity and smoothness properties (e.g., it supports the distance to k^{th} nearest neighbor [78], but not the density-based LOF outliers [15]). According to the proposed algorithm, each node maintains a local list of outliers, along with additional information on the data it has transmitted to its neigh-

bors and the data it has received. Following some rounds of peer-to-peer communications, all the nodes in the network converge to the final list of global outliers. This technique guarantees that it will correctly identify all outliers, but only under the assumptions that each node has accurate knowledge of its nearest neighbors, the communications are reliable, and that the data remains static long enough for the algorithm to converge.

In a similar setting, Zhang et al. [109] describe a technique for identification of global outliers, where outliers are defined as the n points with the largest distance to their k^{th} nearest neighbor. This technique assumes the existence of an aggregation tree, which is used as the communication structure among the nodes in the network. The nodes use the aggregation tree to send local outliers and supporting information to their parents, with the root node eventually collecting all the information. At this point the sink is able to calculate the top-n global outlier candidates, which transmits back to all the nodes in the network for verification. If corrections need to be made, these have to be sent to the sink, which will then adjust the candidate outlier list and repeat the verification process. The end result is guaranteed to be correct as long as the network topology does not change, and the algorithm converges to the solution faster than the data gets updated (which implies the need for a rather slow update rate).

A subsequent study [85] takes a more pragmatic approach, removing the assumptions mentioned in the previous approaches. The goal is still to find global outliers. An outlier is defined as a point whose distance from its k^{th} nearest neighbor is more than a distance threshold d; or alternatively, as a point p, such that there exist no more than n other points with distance to their k^{th} nearest neighbors larger than the distance of p to its k^{th} nearest neighbor. This approach is based on the use of an equi-width histogram that can effectively aggregate and summarize the sensor data readings. The histogram is built in the sink, after the sink agrees with all the sensor nodes on the boundaries of the histogram and its buckets. The histogram is then used by the sink in order to prune the search space of outliers, by eliminating all points that cannot possibly be outliers, as well as identifying points that are certainly outliers. For the points for which no definite answer can be given, the sink will explicitly ask the sensor nodes in the network, in an additional round of computations.

Burdakis et al. [17] present an outlier detection framework that can provide hard guarantees on the results. It is based on the Geometric Approach [81], which allows the development of much more efficient methods (in terms of communication cost) than the ones presented above. The Geometric Approach enables the monitoring of complex (poten-

tially non-linear) functions computed over the average of vectors the describe the local behavior at each sensor node, and the handling of different similarity functions (useful for the outlier detection task) in the distributed setting of a WSN: each sensor is assigned a zone, which is locally monitored, and if no sensor identifies a threshold violation in their corresponding zones, then the overall monitored function will not have exceeded the threshold either. Under the proposed framework, we can identify sensor nodes that involve sensed data values (either the recent history of readings, or the vector of the currently sensed values) that are not similar to the corresponding values of other similar nodes in the network. Several different similarity measures can be efficiently supported, including L_1, L_2, L_∞, cosine similarity, extended Jaccard coefficient, and correlation coefficient.

3.4 Processing Uncertain Data Series

In several different domains, such as manufacturing plants and engineering facilities, sensor networks are being deployed to ensure efficiency, product quality and safety [57]: unexpected vibration patterns in production machines, or changes in the composition of chemicals in industrial processes, are used to identify in advance possible failures, suggesting repairs or replacements. However, sensor readings are inherently imprecise because of the noise introduced by the equipment itself [18].

Previous work has shown that treating value uncertainty as a first class citizen can lead to better results in terms of quality and efficiency [57, 91, 94, 96]. Since value uncertainty is inherent in WSN data, in the following paragraphs we discuss some recent works on processing data series with uncertain values. The focus of these works is on similarity matching, which serves as the basis for developing various more complex analysis and mining algorithms (e.g., classification, clustering, outlier detection, etc.).

Two main approaches have emerged for modeling uncertain data series. In the first, a Probability Density Function (PDF) over the uncertain values is estimated by using some a priori knowledge [112, 105, 83]. In the second, the uncertain data distribution is summarized by repeated measurements (i.e., samples) [8]. We discuss those in more detail below.

3.4.1 Similarity Matching for Uncertain Data Series.

Formally, an uncertain data series T is defined as a sequence of random variables $< t_1, t_2, ..., t_n >$, where t_i is the random variable modeling the real valued number at timestamp i. All the three models we review and compare fit under this general definition.

The problem of similarity matching has been extensively studied in the past [5, 35, 53, 22]: given a user-supplied query sequence, a similarity search returns the most similar data series according to some distance function. More formally, given a collection of data series $C = \{S_1, ..., S_N\}$, where N is the number of data series, we are interested in evaluating the range query function $RQ(Q, C, \epsilon)$:

$$RQ(Q, C, \epsilon) = \{S | S \in C \wedge distance(Q, S) \leq \epsilon\} \tag{7.1}$$

In the above equation, ϵ is a user-defined distance threshold. A survey of representation and distance measures for data series can be found elsewhere [33].

A similar problem arises also in the case of uncertain data series, and the problem of probabilistic similarity matching has been introduced in the last years. Formally, given a collection of uncertain data series $C = \{T_1, ..., T_N\}$, we are interested in evaluation the probabilistic range query function $PRQ(Q, C, \epsilon, \tau)$:

$$PRQ(Q, C, \epsilon, \tau) = \{T | T \in C | Pr(distance(Q, S) \leq \epsilon) \geq \tau\} \tag{7.2}$$

In the above equation, ϵ and τ are the user-defined distance threshold and the probabilistic threshold, respectively.

In the recent years three techniques have been proposed to evaluate PRQ queries, namely MUNICH[7] [8], PROUD [105], and DUST [83]. We discuss each one of these three techniques below, and offer some insights in Section 4.2.

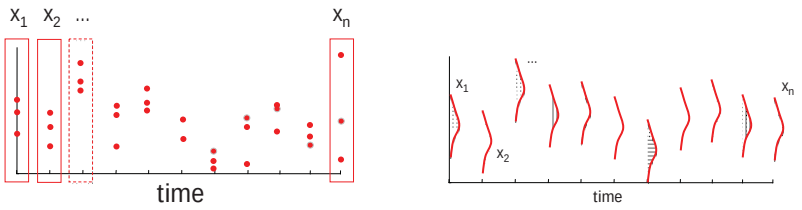
3.4.2 Proposed Techniques. MUNICH: In [8], uncertainty is modeled by means of repeated observations at each timestamp, as depicted in Figure 7.5(a). Assuming two uncertain data series, X and Y, MUNICH proceeds as follows. First, the two uncertain sequences X, Y are materialized to all possible certain sequences: $TS_X = \{< v_{11}, ..., v_{n1} >, ..., < v_{1s}, ..., v_{ns} >\}$ (where v_{ij} is the j-th observation in timestamp i), and similarly for Y with TS_Y. The set of all possible distances between X and Y is then defined as follows:

$$dists(X, Y) = \{L^p(x, y) | x \in TS_X, y \in TS_Y\} \tag{7.3}$$

The uncertain L^p distance is formulated by means of counting the feasible distances:

[7] We will refer to this method as *MUNICH* (it was not explicitly named in the original paper), since all the authors were affiliated with the University of Munich.

$$Pr(distance(X,Y) \leq \epsilon) = \frac{|\{d \in dists(X,Y) | d \leq \epsilon\}|}{|dists(X,Y)|} \quad (7.4)$$



(a) (b)

Figure 7.5. Example of an uncertain data series $X = \{x_1, ..., x_n\}$ [27], modeled by means of repeated observations (a), and *pdf* estimation (b).

Once we compute this probability, we can determine the result set of PRQs similarity queries by filtering all uncertain sequences using Equation 7.4. Note that the naive computation of the result set is infeasible, because of the exponential computational cost, $|dists(X,Y)| = s_X^n s_X^n$, where s_X, s_Y are the number of samples at each timestamp of X, Y, respectively, and n is the length of the sequences. Efficiency can be ensured by upper and lower bounding the distances, and summarizing the repeated samples using minimal bounding intervals [8]. This framework has been applied to Euclidean and DTW distances and guarantees no false dismissals in the original space.

PROUD: In [105], an approach for processing queries over PRObabilistic Uncertain Data streams (PROUD) is presented. Inspired by the Euclidean distance, the PROUD distance is modeled as the sum of the differences of the streaming data series random variables, where each random variable represents the uncertainty of the value in the corresponding timestamp (see Figure 7.5(b)). Given two uncertain data series X, Y, their distance is defined as:

$$distance(X,Y) = \sum_i {D_i}^2 \quad (7.5)$$

where $D_i = (x_i - y_i)$ are random variables, as shown in Figure 7.6.

According to the central limit theorem, we have that the cumulative distribution of the distances approaches a normal distribution, and the normalized distance follows a standard normal distribution. Therefore, we can obtain the normal distribution of the original distance as follows:

$$distance(X,Y) \propto N(\sum_i E[D_i^2], \sum_i Var[D_i^2]) \quad (7.6)$$

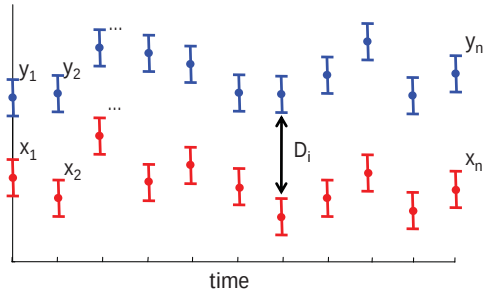


Figure 7.6. The probabilistic distance model [27].

The interesting result here is that, regardless of the data distribution of the random variables composing the uncertain data series, the cumulative distribution of their distances (1) is defined similarly to their euclidean distance and (2) approaches a normal distribution. Recall that we want to answer PRQs similarity queries. First, given a probability threshold τ and the Cumulative Distribution Function (CDF) of the normal distribution, we compute ϵ_{limit} such that:

$$Pr(distance(X,Y)_{norm} \leq \epsilon_{limit}) \geq \tau \tag{7.7}$$

The CDF of the normal distribution can be formulated in terms of the well known *error-function*, and ϵ_{limit} can be determined by looking up the statistics tables. Once we have ϵ_{limit}, we proceed by computing also the normalized ϵ_{norm}. Then, if a candidate uncertain series Y satisfies the inequality $\epsilon_{norm}(X,Y) \geq \epsilon_{limit}$, the following equation holds:

$$Pr(distance(X,Y)_{norm} \leq \epsilon_{norm}(X,Y)) \geq \tau \tag{7.8}$$

Therefore, Y can be added to the result set. Otherwise, it is pruned away. This distance formulation is statistically sound and only requires knowledge of the general characteristics of the data distribution, namely, its mean and variance.

DUST: In [83], the authors propose a new distance measure, DUST, that compared to MUNICH, does not depend on the existence of multiple observations and is computationally more efficient. Similarly to [105], DUST is inspired by the Euclidean distance, but works under the assumption that all the data series values follow some specific distribution. Given two uncertain data series X, Y, the distance between two uncertain values x_i, y_i is defined as the distance between their true (unknown) values $r(x_i), r(y_i)$: $dist(x_i, y_i) = L^1(r(x_i), r(y_i))$. This distance

can then be used to define a function ϕ that measures the similarity of two uncertain values:

$$\phi(|x_i - y_i|) = Pr(dist(0, |r(x_i) - r(y_i)|) = 0) \tag{7.9}$$

This basic similarity function is then used inside the dissimilarity function between two uncertain data values x and y, and we have $dust(x, y) = \sqrt{-\log(\phi(|x - y|)) - k}$, where $k = -\log(\phi(0))$, and for entire uncertain sequences takes the following form:

$$DUST(X, Y) = \sqrt{\sum_i dust(x_i, y_i)^2} \tag{7.10}$$

The handling of uncertainty has been isolated inside the ϕ function, and its evaluation requires to know exactly the data distribution. In contrast to the techniques we reviewed earlier, the DUST distance is a real number that measures the dissimilarity between uncertain data series. Thus, it can be used in the place of the existing distance function in mining techniques that have been developed for certain data series.

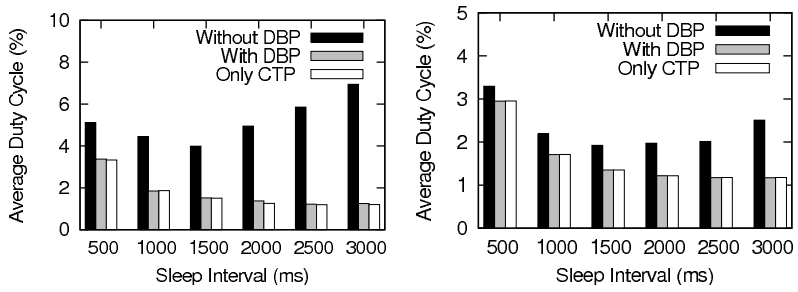
4. Discussion

In this section, we offer some insights on the approaches and techniques we described earlier. This discussion is also useful for determining promising future research directions.

4.1 Data-Aware Network Protocols

In Section 2, we described several techniques for the efficient collection of the sensed data in a WSN. All these techniques invariably claim considerable savings in terms of required communication messages. Experiments have demonstrated savings of up to $2-3$ orders of magnitude, which is very promising news for the energy savings as well, and consequently the lifetime of the WSN. However, these works have not undertaken a careful study of how the communication savings translate to network lifetime prolongation in real deployments.

A recent study [79] focused on exactly this problem: it examined how DBP (similar results can be obtained for other data-driven data acquisition techniques, as well) affected the WSN lifetime, motivated by a real-world WSN-based application deployment in an operational road tunnel. The performance of DBP was studied in conjunction with the commonly-used network stack composed of CTP [42], BoX-MAC [65], and TinyOS v2.1.1. The experimental evaluation used two settings: an operational road tunnel, and an indoor testbed (fed with the same real



(a) Testbed (2 hours). (b) Tunnel (23 hours).

Figure 7.7. Average duty cycle [79]. (Note the difference in the y-axis scale.)

data), representative of scenarios with different connectivity. Based on a 47-day, 40-node dataset gathered in this deployment, the study shows that DBP suppresses 99% of the message reports (w.r.t. the baseline, where all nodes send data every 30 sec).

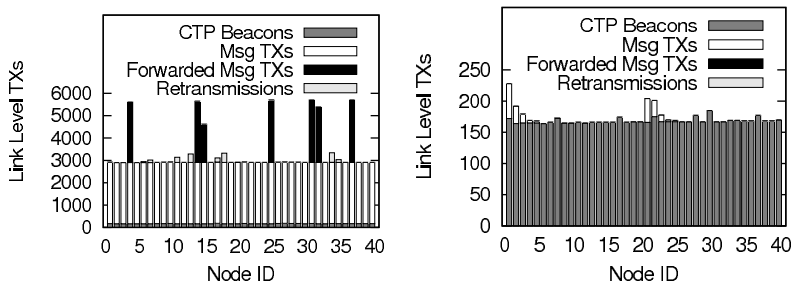
This study examined how data delivery to the application, network lifetime, and routing costs are affected by DBP. To study the impact on lifetime, the study measured the duty cycle of the radio, which is the most power-consuming component. Figure 7.7 clearly shows that DBP enables significant savings at any sleep interval, while the best sleep interval without DBP is 1500 ms . When using DBP, longer sleep intervals can be used to increase lifetime without affecting data delivery.

Figure 7.7(a) shows that in the testbed, with a sleep interval of 1500 ms the WSN running DBP lasts twice as long as with no DBP (with the same MAC settings). Using the best sleep interval in both cases (i.e., 1500 and 3000 ms, respectively) yields a three-fold lifetime improvement[8].

A natural question arises at this point: if DBP suppresses over 99% of the messages, why does the network lifetime increase "only" three-fold? This is due to the costs of the network stack, in particular the idle listening and average transmission times of the MAC protocol, and to the overhead of the routing protocol to build and maintain the data collection tree.

To isolate the inherent costs (e.g., tree maintenance) of CTP, experiments were ran with no application traffic. Figure 7.7 shows the corresponding duty cycle (as *Only CTP*). We observe that the DBP cost is

[8]The energy savings in the tunnel (see Figure 7.7(b)) are less remarkable, although still significant, because the network diameter in the tunnel is much smaller w.r.t. the testbed (due to the waveguide effect [66] many direct, 1-hop links to the sink exist, leaving less room for improvement).



(a) Without DBP. (b) With DBP.

Figure 7.8. Tunnel: total link-level transmissions for a sleep interval of 1500 ms [79]. (Note the difference in the y-axis scale.)

very close to the cost of CTP tree maintenance, regardless of the sleep interval. A finer-grained view is provided by Figure 7.8, which shows the different components of traffic in the network. Without DBP, the dominate component is message transmission and forwarding; significant retransmissions are present for some nodes, while the component ascribed to CTP (i.e., the beacons probing for link quality) is negligible. When DBP is active, the number of CTP beacons remains basically unchanged. However, because the application-level traffic is dramatically reduced, CTP beacons become the dominant component of the network traffic.

These last observations suggest that further reductions in data traffic would have little practical impact on the system lifetime, as routing costs are dominated by topology maintenance rather than data forwarding. Therefore, improvements are more likely to come from radical changes at the routing and MAC layers: new, data-aware protocols need to be designed, which will take into account the traffic patterns with extremely low data rates that emerge when data-driven data acquisition techniques are employed.

4.2 Uncertain Data Processing

Given the fact that sensors produce values with an inherent uncertainty, and that we are increasingly relying on applications that are driven by sensor data, it becomes evident that efficient and effective processing of uncertain WSN data series is a relevant research direction.

Turning our attention to the three techniques we presented for uncertain data series similarity matching (see Section 3.4), we observe that an important factor for choosing among them is the information

that is available about the distribution of the data series and its errors. PROUD requires to know the standard deviation of the uncertainty error and a single observed value for each timestamp, and assumes that the standard deviation of the uncertainty error remains constant across all timestamps. DUST takes as input a single observed value of the data series for each timestamp, and in addition, needs to know the distribution of the uncertainty error for each single time stamp, as well as the distribution of the values of the data series. This means that, in contrast to PROUD, DUST can take into account mixed distributions for the uncertainty errors (albeit, they have to be explicitly provided in the input). MUNICH does not need to know the distribution of the data series values, or the distribution of the value errors: it simply operates on the observations available at each timestamp. When we do not have enough, or accurate information on the distribution of the errors, PROUD and DUST do not offer an advantage in terms of accuracy when compared to Euclidean [27].

All three techniques are based on the simplifying assumption that the values of the data series are independent from one another, which is not true for WSN measurements. A recent study [28] demonstrates that removing this assumption is beneficial: it proposes the UMA and UEMA filters (based on the weighted moving average technique), that in combination with Euclidean distance lead to more accurate results. These results suggest that more work is needed on techniques that take into account the temporal correlations that exist in data series.

The time complexity of these techniques is another important factor. We note that MUNICH is applicable only in the cases where the standard deviation of the error is relatively small, and the length of the data series is also small (otherwise the computational cost is prohibitive), which makes this technique applicable in cases where the sink can do the processing. To a (much) lesser extent, this is also true for PROUD and DUST. On the other hand, UMA and UEMA have significantly lower resource requirements, and could be efficiently implemented in a sensor node.

4.3 Ubiquitous Sensor Networks

Lots of work and research effort has been devoted in the past years in the study of various problems related to WSNs. Several efficient techniques have been developed for the acquisition, management, processing, and analysis of the sensed data, and at the same time (different forms of) WSNs are being deployed in increasingly more domains and situations.

The next frontier in this line of research is the development of very large, ubiquitous WSNs, with increased capabilities for complex, in-network analytics. This vision includes various wireless devices with different specifications (ranging from simple sensor motes to state of the art smartphones), involves advanced, yet efficient, data management and processing techniques, and calls for new breakthroughs in several of the problems and research directions we discussed in the previous sections.

Consider a large WSN deployment, such as *SmartSantander* [2], which comprises of more than 20,000 sensors in an urban setting. This system has already started to be installed, and can drive the development of powerful applications with a big environmental and societal impact (e.g., environment-aware traffic and transportation monitoring and management, where traffic is managed in real-time, according to levels of pollutants, noise, local events, emergency situations, etc.).

As these WSNs grow larger, covering more space and involving more devices, it makes sense to increase their ability to ingest and process more data in real-time, and to run complex queries in a distributed manner more effectively. This will allow large numbers of queries to run within the WSN, sharing and exchanging results, and with the goal to minimize the need for centralized processing and human intervention (or opportunistically seek human intervention, as in crowdsourcing environments). In order to achieve these goals, methodologies and techniques from other domains could be exploited and adapted (apart from what we have already described here), such as distributed complex event processing [14, 95], and distributed publish/subscribe systems [37].

5. Conclusions

The development of WSN during the past decade helped advance the state of art in several scientific communities that exploited the new opportunities for fine-granularity data-gathering. The popularity of WSNs has also provoked the interest of the research community, and a multitude of studies have been published on techniques and methodologies for the effective and efficient use of the data produced by WSNs, across the networks and data management communities.

As we are now going through the second decade of the WSNs lifetime, we are witnessing a widening and increasing interest in their potential applications, finding their way in new domains and also including new types of devices (e.g., smartphones). In this context, old problems re-emerge, such as the design of novel network protocols that are data-aware, and new challenging problems appear, such as the effective management of

uncertainty in sensed data series, and techniques that will scale the in-network complex analytics to very large WSNs.

Acknowledgments

I would like to acknowledge the contributions of my colleagues and collaborators: Alessandro Camerra, Michele Dallachiesa, Adriola Faqolli, Dimitrios Gunopulos, Marwan Hassani, Vana Kalogeraki, Eamonn Keogh, Usman Raza, Katsiaryna Mirylenka, Emmanuel Muller, Amy L. Murphy, Besmira Nushi, Dimitris Papadopoulos, Gian Pietro Picco, Thomas Seidl, Pascal Spaus, Sharmila Subramaniam, Wagner Trupel, Paolo Valleri, and Michail Vlachos.

References

[1] Pacific Northwest Weather Data. `http://www-k12.atmos.washington.edu/k12/grayskies/`, 2012.

[2] SmartSantander: A City-wide Experimental Facility. `http://www.smartsantander.eu/`, 2012.

[3] C. C. Aggarwal, A. Bar-Noy, and S. Shamoun. On sensor selection in linked information networks. In *DCOSS*, pages 1–8, 2011.

[4] C. C. Aggarwal, Y. Xie, and P. S. Yu. On dynamic data-driven selection of sensor streams. In *KDD*, pages 1226–1234, 2011.

[5] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[6] A. Ali, A. Khelil, F. Karim Shaikh, and N. Suri. MPM: Map based predictive monitoring for wireless sensor networks. In *Proc. of Int. Conf. on Autonomic Computing and Communication Systems*, 2009.

[7] M. H. Ali, M. F. Mokbel, W. G. Aref, and I. Kamel. Detection and tracking of discrete phenomena in sensor-network databases. In *SSDBM*, pages 163–172, Santa Barbara, CA, 2005.

[8] J. Aßfalg, H.-P. Kriegel, P. Kröger, and M. Renz. Probabilistic similarity search for uncertain time series. In *SSDBM*, pages 435–443, 2009.

[9] E. Baralis and T. Cerquitelli. Selecting representatives in a sensor network. In *In Proceedings of the SEBD*, pages 351–360, 2006.

[10] A. Barreto, A. Araujo, and S. Kremer. A taxonomy for spatiotemporal connectionist networks revisited: the unsupervised case. *Neural Computation*, 15:1255–1320, 2003.

[11] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.

[12] L. M. A. Bettencourt, A. A. Hagberg, and L. B. Larkey. Separating the wheat from the chaff: Practical anomaly detection schemes in ecological applications of distributed sensor networks. In *DCOSS*, pages 223–239, 2007.

[13] J. W. Branch, B. K. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *ICDCS*, page 51, 2006.

[14] L. Brenna, J. Gehrke, M. Hong, and D. Johansen. Distributed event stream processing with non-deterministic finite automata. In *Proc. of ACM Conference on Distributed Event-Based Systems (DEBS)*, pages 1–12, 2009.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD Conference*, pages 93–104, 2000.

[16] A. Bulut and A. K. Singh. SWAT: Hierarchical Stream Summarization in Large Networks. In *International Conference on Data Engineering*, pages 303–314, Bangalore, India, March 2003.

[17] S. Burdakis and A. Deligiannakis. Detecting outliers in sensor networks using the geometric approach. In *ICDE*, 2012.

[18] M. Ceriotti, M. Corra, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. Lo Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele. Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels. In *International Conference on Information Processing in Sensor Networks (IPSN)*, pages 187–198, 2011.

[19] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corrà, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *IPSN*, pages 277–288, 2009.

[20] K. Chakrabarti, E. J. Keogh, S. Mehrotra, and M. J. Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[21] K. Chan and W. Fu. Efficient Time Series Matching by Wavelets. In *International Conference on Data Engineering*, pages 126–133, Sydney, Australia, March 1999.

[22] K.P. Chan and A.W.C. Fu. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133. IEEE, 2002.

[23] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[24] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad-hoc Networks Journal*, 2003.

[25] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Proc. of the Int. Conf. on Data Eng. (ICDE)*, 2006.

[26] R. Cole, D. Shasha, and X. Zhao. Fast window correlations over uncooperative time series. In *KDD*, pages 743–749, 2005.

[27] M. Dallachiesa, B. Nushi, T. Palpanas, and K. Mirylenka. Similarity matching for uncertain time series: analytical and experimental comparison. In *QUeST*, 2011.

[28] M. Dallachiesa, B. Nushi, T. Palpanas, and K. Mirylenka. Uncertain time series similarity: Return to the basics. In *VLDB*, 2012.

[29] B. de Vries and J. C. Principe. The gamma model — A new neural model for temporal processing. *Neural Networks*, 5:565–576, 1992.

[30] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another outlier bites the dust: Computing meaningful aggregates in sensor networks. In *ICDE*, pages 988–999, 2009.

[31] A. Deshpande, C. Guestrin, and S. Madden. Using probabilistic models for data management in acquisitional environments. In *CIDR*, pages 317–328, 2005.

[32] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *VLDB*, Toronto, ON, Canada, 2004.

[33] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[34] E. Elnahrawy and B. Nath. Context-aware sensors. In *EWSN*, pages 77–93, 2004.

[35] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *ACM SIGMOD Record*, 23(2):419–429, 1994.

[36] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. In *ACM SIGMOD*

International Conference, pages 419–429, Minneapolis, MI, USA, May 1994.

[37] E. Fidler, H. a. Jacobsen, G. Li, and S. Mankovski. The padres distributed publish/subscribe system. In *Proc. of the Conference on Feature Interactions in Telecommunications and Software Systems*, 2005.

[38] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan. Multiresolution storage and search in sensor networks. *ACM TOS*, 1(3):27–315, 2005.

[39] N. Giatrakos, Y. Kotidis, and A. Deligiannakis. Pao: power-efficient attribution of outliers in wireless sensor networks. In *DMSN*, pages 33–38, 2010.

[40] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. Taco: tunable approximate computation of outliers in wireless sensor networks. In *SIGMOD Conference*, pages 279–290, 2010.

[41] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, pages 79–88, 2001.

[42] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. The collection tree protocol. In *Proc. of the Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2009.

[43] D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. In *IPSN*, pages 220–231, 2010.

[44] L. Gruenwald, Md. S. Sadik, R. Shukla, and H. Yang. DEMS: A Data Mining Based Technique to Handle Missing Data in Mobile Sensor Network Applications. In *Proc. of the Int. Conf. on Data Mgmt. for Sensor Networks (DMSN)*, 2010.

[45] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. In *IPSN*, Berkeley, CA, 2004.

[46] M. Hassani, E. Müller, and T. Seidl. EDISKCO: Energy efficient distributed in-sensor-network k-center clustering with outliers. In *In Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data SensorKDD*, pages 39–48, 2009.

[47] M. Hassani, E. Müller, P. Spaus, A. Faqolli, T. Palpanas, and T. Seidl. Self-organizing energy aware clustering of nodes in sensor networks using relevant attributes. In *Proc. of the Int. Wkshp. on Kowledge Discovery from Sensor Data (SensorKDD)*, 2010.

[48] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[49] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, pages 63–79, 2003.

[50] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS*, 2002.

[51] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using Kalman filters. In *Proc. of the Int. Conf. on Management of Data (SIGMOD)*, 2004.

[52] H. Jiang, S. Jin, and C. Wang. Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Trans. on Parallel Distributed Systems*, 22, June 2011.

[53] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[54] E. J. Keogh and M. J. Pazzani. An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *International Conference on Knowledge Discovery and Data Mining*, pages 239–243, New York, NY, USA, August 1998.

[55] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*, NY, NY, 1998.

[56] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *Proceeding of the 21st International Conference on Data Engineering, ICDE*, 2005.

[57] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 64–75. ACM, 2005.

[58] I. Lazaridis and S. Mehrotra. Capturing Sensor-Generated Time Series with Quality Guarantees. In *International Conference on Data Engineering*, pages 429–440, Bangalore, India, March 2003.

[59] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. of the Int. Conf. on Data Eng. (ICDE)*, 2002.

[60] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[61] N. Malpani, J. Welch, and N. Vaidya. Leader Election Algorithms for Mobile Ad Hoc Networks. In *Proc. Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2000.

[62] A. Meka and A. K. Singh. Distributed special clustering in sensor networks. In *EDBT 2006, LNCS 3896*, pages 980–1000, 2006.

[63] R. A. F. Mini, M. Do Val Machado, A. A. F. Loureiro, and B. Nath. Prediction-based energy map for wireless sensor networks. *Ad Hoc Networks*, 3, 2005.

[64] M. Moshtaghi, C. Leckie, S. Karunasekera, J. C. Bezdek, S. Rajasegarar, and M. Palaniswami. Incremental elliptical boundary estimation for anomaly detection in wireless sensor networks. In *ICDM*, pages 467–476, 2011.

[65] D. Moss and P. Levis. BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking. Technical Report SING-08-00, 2008.

[66] L. Mottola, G.P. Picco, M. Ceriotti, S. Guna, and A.L. Murphy. Not All Wireless Sensor Networks Are Created Equal: A Comparative Study On Tunnels. *ACM Trans. on Sensor Networks (TOSN)*, 7(2), 2010.

[67] S. Nath. Energy efficient sensor data logging with amnesic flash storage. In *IPSN*, pages 157–168, 2009.

[68] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *IPSN*, pages 80–95, Palo Alto, CA, 2003.

[69] T. Palpanas, V. Kalogeraki, and D. Gunopulos. Online distribution estimation for streaming data: Framework and applications. In *SEBD*, pages 430–438, 2007.

[70] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos. Streaming time series summarization using user-defined amnesic functions. *IEEE Trans. Knowl. Data Eng.*, 20(7):992–1006, 2008.

[71] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral, 2003.

[72] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[73] N. D. Pham, T. D. Le, and H. Choo. SCCS: Spatiotemporal clustering and compressing schemes for efficient data collection applications in WSNs. *Int. Journal of Communication Systems*, 23, 2010.

[74] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. of the Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2005.

[75] I. Popivanov and R. J. Miller. Similarity Search Over Time Series Data Using Wavelets. In *International Conference on Data Engineering*, pages 802–813, San Jose, CA, USA, February 2002.

[76] D. Rafiei. On Similarity-Based Queries for Time Series Data. In *International Conference on Data Engineering*, Sydney, Australia, March 1999.

[77] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek. Quarter sphere based distributed anomaly detection in wireless sensor networks. In *ICC*, pages 3864–3869, 2007.

[78] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD Conference*, pages 427–438, 2000.

[79] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. What does model-driven data acquisition really achieve in wireless sensor networks? In *IEEE International Conference on Pervasive Computing and Communications*, Lugano, Switzerland, 2012.

[80] P. P. Rodrigues, J. Gama, and L. Lopes. Clustering distributed sensor data streams. In *ECML PKDD 2008, LNAI. Springer-Verlag*, 2008.

[81] G. Sagy, D. Keren, I. Sharfman, and A. Schuster. Distributed threshold querying of general functions by a difference of monotonic representation. *PVLDB*, 4(2):46–57, 2010.

[82] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD Conference*, pages 599–610, 2005.

[83] S.R. Sarangi and K. Murthy. DUST: a generalized notion of similarity between uncertain time series. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–392. ACM, 2010.

[84] D. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley & Sons, 1992.

[85] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *MobiHoc*, pages 219–228, 2007.

[86] A. Silberstein, R. Braynard, and J. Yang. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In *SIGMOD Conference*, pages 157–168, 2006.

[87] A. Silberstein, G. Filpus, K. Munagala, and J. Yang. Data-driven processing in sensor networks. In *Proc. of the Conf. on Innovative Data Systems Research (CIDR)*, 2007.

[88] A. Silberstein, A. E. Gelfand, K. Munagala, G. Puggioni, and J. Yang. Making sense of suppressions and failures in sensor data: A bayesian approach. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, 2007.

[89] E. Soroush, K. Wu, and J. Pei. Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In *MobiHoc*, pages 391–400, 2008.

[90] D. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole. Research Challenges in Environmental Observation and Forecasting Systems. In *Mobile Computing and Networking*, Boston, MA, USA, August 2000.

[91] M. Stonebraker, J. Becla, D. J. DeWitt, K.-T. Lim, D. Maier, O. Ratzesberger, and S. B. Zdonik. Requirements for science data bases and scidb. In *CIDR*, 2009.

[92] S. Subramaniam, V. Kalogeraki, and T. Palpanas. Distributed Real-Time Detection and Tracking of Homogeneous Regions in Sensor Networks. In *RTSS*, Rio de Janeiro, Brazil, 2006.

[93] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, Seoul, Korea, 2006.

[94] D. Suciu, A. Connolly, and B. Howe. Embracing uncertainty in large-scale computational astrophysics. In *MUD*, pages 63–77, 2009.

[95] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*, chapter 13, pages 603–607. Prentice Hall, 2006.

[96] T. T. L. Tran, L. Peng, B. Li, Y. Diao, and A. Liu. Pods: a new model and processing algorithms for uncertain data streams. In *SIGMOD Conference*, pages 159–170, 2010.

[97] D. Tulone and S. Madden. An energy-efficient querying framework in sensor networks for detecting node similarities. In *Proc. of the*

Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2006.

[98] D. Tulone and S. Madden. PAQ: Time series forecasting for approximate query answering in sensor networks. In *Proceedings of the European Wkshp. on Wireless Sensor Networks (EWSN)*, 2006.

[99] M. C. Vuran, O. B. Akan, and I. F. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3), 2004.

[100] B. Warneke, M. Last, B. Liebowitz, and K. Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer Magazine*, pages 44–51, January 2001.

[101] W. Wu, H.-B. Lim, and K.-L. Tan. Query-driven data collection and data forwarding in intermittently connected mobile sensor networks. In *Proc. of Int. Conf. on Data Mgmt. for Sensor Networks (DMSN)*, 2010.

[102] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng. Localized outlying and boundary data detection in sensor networks. *IEEE Trans. Knowl. Data Eng.*, 19(8):1145–1157, 2007.

[103] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proc. of the Conf. on Innovative Data Systems Research (CIDR)*, 2003.

[104] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks. In *MOBICOM*, Atlanta, GA, USA, 2002.

[105] M.Y. Yeh, K.L. Wu, P.S. Yu, and M.S. Chen. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 684–695. ACM, 2009.

[106] B. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary LP-Norms. In *VLDB International Conference*, pages 385–394, Cairo, Egypt, September 2000.

[107] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE*, pages 13–22, 2000.

[108] J. Yin and M. M. Gaber. Clustering distributed time series in sensor networks. In *In Proceedings of the Eighth IEEE Conference on Data Mining, ICDM*, 2008.

[109] K. Zhang, S. Shi, H. Gao, and J. Li. Unsupervised outlier detection in sensor networks using aggregation tree. In *ADMA*, pages 158–169, 2007.

[110] S. Zhao, K. Tepe, I. Seskar, and D. Raychaudhuri. Routing protocols for self-organizing hierarchical ad hoc wireless networks. In *IEEE Sarnoff Symposium*, 2003.

[111] Y. Zhao and S. Zhang. Generalized dimension-reduction framework for recent-biased time series analysis. *IEEE Trans. Knowl. Data Eng.*, 18(2):231–244, 2006.

[112] Y. Zhao, C. C. Aggarwal, and P. S. Yu. On wavelet decomposition of uncertain time series data sets. In *CIKM*, pages 129–138, 2010.

[113] Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In *Proc. of the Int. Conf. on Computer Communications and Networks (IC3N)*, 2004.

[114] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.

[115] Y. Zhuang and L. Chen. In-network outlier cleaning for data collection in sensor networks. In *CleanDB*, pages 41–48, 2006.

Chapter 8

DISTRIBUTED DATA MINING IN SENSOR NETWORKS

Kanishka Bhaduri
Netflix Inc.
100 Winchester Circle
Los Gatos, CA 94032
kanishka.bh@gmail.com

Marco Stolpe
TU Dortmund University
Artificial Intelligence Group, LS 8
Joseph-von-Fraunhofer-Straße 23
44227, Dortmund, Germany
marco.stolpe@tu-dortmund.de

Abstract Wireless sensor networks (WSNs) consist of a collection of low cost and low powered sensor devices capable of communicating with each other via an ad-hoc wireless network. Due to their rapid proliferation, sensor networks are currently used in a plethora of applications such as earth sciences, systems health, military applications etc. These sensors collect the data about the environment and this data can be mined for a variety of analysis. Unfortunately, post analysis of the data extracted from the WSN incurs high sensor communication cost for sending the raw data to the base station and at the same time runs the risk of delayed analysis. To overcome this, researchers have proposed several distributed algorithms which can deal with the data in situ – these data mining algorithms utilize the computing power at each node to first do some local computations and then exchange messages with its neighbors to come to a consensus regarding a global model. These algorithms reduce the communication cost vastly and also are extremely efficient in terms of model computation and event detection. In this chapter we focus on such distributed data mining algorithms for data clustering, classification and outlier detection tasks.

Keywords: distributed data mining, sensor networks, outlier detection

1. Introduction

A wireless sensor network (WSN) [17] consists of a collection of sensors or nodes capable of monitoring the environment using its local sensors and by wirelessly communicating with other nodes and to a base station. WSNs may vary widely in their topology from simple star or ring network to complicated multi-hop networks. Each node is designed to work autonomously using its own battery power. Due to limited battery power, the nodes are constrained in terms of sensing capability, computational power and transmission ability. Their major task is to monitor an environment for a long period of time and hence conserving battery power by turning off the transmission channels is one of the crucial techniques that need to be used for algorithm development and deployment in such networks. Over the last decade, the sensor nodes have evolved a lot in terms of their size and sensing/transmission capability. As a result, there has been a renewed interest in using sensor networks for a plethora of applications – forest fire detection, air pollution monitoring, oceanographic applications, system health monitoring, greenhouse monitoring, battlefield and other military applications to name a few. The main characteristics of a WSN are:

- Limited computation and transmission ability
- Frequent and recurrent node failures
- Unreliable communication links
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions

Given these constraints, it is easy to see than the standard data mining/machine learning algorithms are not directly applicable to a WSN setting. As a result, researchers have proposed several algorithms for modern sensor networks which take into account some or all of these constraints. One of the main items to consider for WSN is reduce the sensor communication requirements for broadcasting all the data to the base station and, it is in this context, that distributed data mining is likely to play a major role. The major goal of such distributed algorithms is to develop methods so that a node first does some local computation on its own data, and then communicates with nearby neighbors (in-network processing) to compute a global model. In this chapter, we present a sampling of three important topics of distributed data mining

algorithms for WSN. These are data and node clustering, data classification and outlier detection in sensor networks.

The rest of this chapter is organized as follows. Section 2 discusses several WSN clustering strategies including both node and data clustering. In the next section (Section 3) we discuss classification techniques followed by several outlier detection techniques in Section 4. We conclude the chapter in Section 5.

2. Clustering in Wireless Sensor Networks

Cluster analysis is the unsupervised learning task of dividing a set of objects into groups (*clusters*), such that a given quality criterion is maximized. More formally, given a set of objects $X = \{x_1, \ldots, x_n\}$, we search for a *clustering* $\mathcal{C} = \{C_1, \ldots, C_k\}$ with $C_i \subseteq X$ for $i = 1, \ldots, k$ that maximizes a quality function $q : 2^X \to \mathbb{R}_0^+$. Partitional clustering algorithms yield disjunct clusters, *i.e.* $C_i \cap C_j = \emptyset$ for $i \neq j$, while the clusters produced by non-partitional algorithms may overlap. Moreover, hierarchical clustering algorithms can subdivide clusters further, resulting in increasingly more detailed groupings of the given objects.

Instead of creating new quality functions and algorithms for each particular application, clustering algorithms usually try to achieve more general parameterized objectives. For example, an often stated objective is that objects in the same cluster should be more similar to each other than objects from different clusters. With $d : X \times X \to \mathbb{R}_0^+$ being a given dissimilarity function between objects, this criterion can be formalized as minimizing the *intra-cluster variance*, also called the *sum of squares within* (SSW), over all possible clusterings for a fixed number of clusters k [32]:

$$\min_{\mathcal{C}^*} \quad SSW(\mathcal{C}^*) = \frac{1}{2} \sum_{i=1}^{k} \sum_{x_1 \in C_i} \sum_{x_2 \in C_i} d(x_1, x_2) \quad \text{with} \quad C_i \in \mathcal{C}^*$$

For algorithms that try to minimize this criterion, like the well-known k-Means algorithm [39], an application specific clustering can then be obtained by choosing an appropriate dissimilarity measure. Density-based algorithms, like DBSCAN [24], form clusters of points that have a high spatial density. Subspace clustering algorithms specialize on finding clusters in lower dimensional subspaces of the whole data space. For a good overview of different types of clustering algorithms and their objectives, see Han [30] and Kriegel *et al.* [38].

The aforementioned algorithms assume the whole data set to be in main memory or at least they ignore the time needed for accessing the

data. They also assume an unlimited amount of available energy. Therefore, the algorithms won't work in the highly constrained setting of distributed wireless sensor nodes. For WSNs, the algorithms have been either modified or new clustering algorithms have been developed that take into account the distributed nature of sensors and the severe communication constraints due to limited battery power and bandwidth. As will become apparent, in WSNs usually multiple quality criterions can be applied, turning clustering into a multi-objective optimization problem.

In the next section it is shown that, while often not directly focused on data analysis, clustering algorithms play a crucial rule in creating communication efficient topologies of sensor nodes. Especially, the solutions found for grouping sensor nodes may inspire the design of future energy-constrained data analysis methods. The section afterwards then discusses already existing distributed clustering algorithms for data analysis, *i.e.* of sensor measurements.

2.1 Distributed Clustering of Sensor Nodes

Continuous monitoring as well as intermittent querying of sensor networks involves transmitting data from individual sensor nodes, the *sources*, to a single node, the *sink*. Communication costs increase with higher distance r between sensor nodes, as ground reflections from short antenna heights may cause a drop-off of the radio signal power by r^4 [47]. Therefore, hierarchical, tiered *multi-hop* architectures with shorter distances between relaying nodes are usually more energy-efficient than letting all sensors communicate directly with some base station [25].

The sensor nodes in tiered multi-hop networks form — possibly hierarchical — clusters and certain nodes in each cluster are designated as *cluster heads*. Cluster heads fulfill special roles, like relaying signals from local nodes in their cluster to other cluster heads or a base station. They also can manage and restrict network access as well as the life cycle of local nodes, or reduce the amount of data transmitted by aggregating and pre-processing the signals from sensor nodes in their cluster.

Manual placement of sensors and routing through pre-determined paths are only feasible for very small networks. However, typical applications of sensor networks, like environmental monitoring, disaster management or military surveillance missions envision hundreds or even thousands of sensor nodes [1], possibly deployed randomly, *e.g.* dropped by a helicopter. The network is usually left unattended for long periods of time and batteries can't be recharged. While some setups utilize mobile sensors, sensor nodes are usually assumed to operate stationary after deployment. Nevertheless, the network could change over time, since

battery-operated sensors may run out of energy and harsh environmental conditions can damage network components. In these scenarios, algorithms are needed that cluster sensor nodes and determine cluster heads dynamically, forming the infrastructure in an ad-hoc manner. Also, they must be able to reconfigure the network when necessary.

Clustering algorithms that have been developed for WSNs mainly differ in their assumptions on the given *network components*, the desired *topology*, and in the *goals* they try to achieve. These in turn influence the used methodologies and running times.

Regarding *network components*, clustering can become more constraint in heterogenous networks where cluster heads have a higher capacity than sensor nodes. Here, the available number of high capacity components will determine the maximum number of cluster heads and therefore the number of clusters. Moreover, if communication costs between cluster heads and sensor nodes are to be minimized, a stationary location of cluster heads will lead to a static assignment of sensors to clusters, except for cases where cluster heads fail and the network needs to be reconfigured. In comparison, in more homogenous networks, also regular sensor nodes can become cluster heads. Clustering algorithms for these networks are usually more dynamic, as they need to continuously balance the energy consumption across all nodes, based on their residual energy. Several algorithms achieve this, for instance, by a regular rotation of cluster heads.

The required *topology* is largely dependent on the given distances between sensor nodes, cluster heads and base stations. Depending on the placement of nodes, the network topologies that need to be considered can reach from fixed 1-hop [33] over fixed k-hop [64] to fully adaptive architectures [21]. An important objective is that network components remain connected, *i.e.* that sensor nodes are able to reach their cluster heads and that cluster heads can reach a base station. Other objectives like minimizing the intra-cluster energy-consumption may need to be trade-off against the goal of components staying connected, for example in cases where an energy-optimal cluster head could no longer reach its base station. Taking into account several — possibly contradicting — quality criteria thus turns clustering in WSNs into a multi-objective optimization problem.

The main *goals* that cluster algorithms for WSNs try to achieve are maximal network longevity, connectivity and fault-tolerance. Extending the operational life-time of a WSN requires load-balancing strategies that prevent premature exhaustion of subsets of sensor nodes and cluster heads. The goal of maintaining connectivity is concerned with ensuring that the most important network components can reach each other, pos-

sibly putting constraints on the clustering. Fault-tolerance deals with the failure of network components and can be achieved by redundancy, rotating roles of network components as well as re-clustering.

As a survey article by Abbasi and Younis [1] shows, the clustering algorithms for sensor nodes are quite diverse and hard to categorize. Moreover, there already exist more algorithms than can sufficiently be presented here, even in summary. Therefore, we decided to focus on only two algorithms in more detail. The algorithms were chosen as examples for demonstrating how the same network topology can be achieved by entirely different means and with different running times. At the end of this section, the reader is then pointed to further algorithms.

2.1.1 Hierarchical control clustering. The clustering scheme introduced by Banerjee and Khuller [7] forms a hierarchical multi-hop network topology, where the number of layers is determined automatically. The original problem statement is that given an undirected graph $G = (V, E)$ and a positive integer k with $1 \leq k \leq |V|$, for each connected component clusters $V_1, \ldots, V_l$ with $V_i \subseteq V$ should be found such that (1) all vertices are part of a cluster, (2) all subgraphs induced by V_i are connected, (3) cluster size is bounded by $k \leq |V_i| < 2k$, (4) two clusters should only have few common vertices and (5) each vertex belongs to a constant number of clusters. After demonstrating that requirement (5) could be violated in general graphs, the problem is restricted to *bounded disk graphs*, as they are usually given in WSNs. For $R_{\min}$ and $R_{\max}$ being the minimum and maximum transmission radius over all nodes, (u, v) is an edge in G if and only if $R_{\min} \leq d(u, v) \leq R_{\max}$. The algorithm then guarantees that no node is a member of more than $O(\log(R_{\max}/R_{\min}))$ clusters. Furthermore, to fulfill requirement (4), it is necessary to allow a single cluster in G to have a size smaller than k.

The distributed algorithm consists of two phases: *cluster creation* and *cluster maintenance*. The cluster formation process can be started by an arbitrary node in the network, which becomes the root node of a Breadth-First-Search (BFS) tree. The initiator with the least node ID takes precedence. Every t units of time, each node u broadcasts a *tree discovery* message to nodes that are in its transmission radius. The message contains a source ID, parent ID (initially not set), the ID of the root node, a sequence number and the shortest (known) hop-distance to the root, r. A node v will make u its parent and update its hop-distance if the route through u to r is shorter. The root ID and sequence number are used to distinguish between multiple instances of the cluster creation phase. Next, for *cluster formation*, the sent messages are extended by additional fields representing *subtree size* and *node adjacency*.

The size information is aggregated bottom up. When the subtree size of a node w crosses the size parameter k, it forms clusters on its subtree $T(w)$. If $|T(w)| < 2k$, a single cluster containing $T(w)$ is created. Otherwise, children subtrees will be appropriately partitioned, using the node adjacency information. The cluster assignments are propagated to the relevant nodes by *cluster assignment* messages. Once clusters have been formed for $T(w)$, w does not include information about these nodes in subsequent messages. Nodes send a *terminate cluster* message down their subtrees if subtree sizes have not changed for a fixed amount of time. At the end, only the cluster assignments need to be maintained, while the BFS information is unimportant. During cluster maintenance, a sensor node joining the network may either be assigned to an existing nearby cluster V_i, if $|V_i| < 3k - 1$, or clusters are split, like in the cluster creation phase. If existing nodes leave the network, clusters can become disconnected. However, the number of remaining connected components is bounded, since no node is a member of more than $O(\log(R_{\max}/R_{\min}))$ clusters (see above). The connected components are either made clusters of their own or, if their size is $< k$, their nodes will try to join a neighboring cluster. The same is true in cases of link outages and network partitions.

The algorithm converges in $O(n)$ steps, where n is the number of sensor nodes. In principle, it can work with mobile sensor nodes and recover from network failures. It achieves the self-organization of sensor nodes into a multi-hop network and reduces transmission distances, since parent nodes are chosen by the shortest known hop-distance to the root.

2.1.2 DWEHC. Ding *et al.* [21] have proposed a distributed weight-based energy-efficient hierarchical clustering protocol (DWEHC). The key idea is to elect cluster heads not only based on distances from a node to all its neighbors, but also take into account the residual energy of nodes. A basic observation here is that for devices with similar antenna heights, the transmitter power required by distance r is r^α. For three nodes s, r and d, a direct transmission from s to d takes power $||sd||^\alpha + c$, while relaying transmission through a node r takes power $||sr||^\alpha + c + ||rd||^\alpha + c$. In cases where $||sd||^\alpha + c > ||sr||^\alpha + c + ||rd||^\alpha + c$, relaying is more efficient. The neighbors $N_{\alpha,c}(s)$ of a node s are defined as the set of nodes that lie in the transmission range of s and need no relaying. The weight $W(s)$ is then calculated as

$$W(s) = \left(\sum_{u \in N_{\alpha,c}(s)} \frac{(R-d)}{6R} \right) \times \frac{E_{\text{residiual}}(s)}{E_{\text{initial}}(s)}$$

where R is the cluster range (the farthest distance nodes can be from their cluster heads) and $E_{\text{initial}}(s)$ and $E_{\text{residiual}}(s)$ are the initial and residual energy of node s respectively. The average number of neighboring nodes can be shown to be at most six [21]. Intra-cluster communication will be at minimum when the transmission graph contains the shortest paths between all pairs of nodes in the cluster.

The protocol starts with each node u broadcasting its (x, y) coordinates, establishing its local neighborhood $N_{\alpha,c}(u)$, calculating its weight $W(s)$ and broadcasting it. A node s sets $level(s) = -1$, indicating that it hasn't joined any cluster yet. In the *cluster generation* phase, the following procedure is repeated for a fixed number of six iterations. Let i be the iteration number. A node s first checks if it is assigned to a cluster. If not, it will become a temporary cluster head if its weight is largest among it neighbors, otherwise the neighbor with the largest weight is chosen as a temporary head for s. The ID of the temporary head is broadcasted to all neighbors of s. A node becomes a real cluster head only if a percentage of $(6 - i)/6$ nodes elect the node as their temporary cluster head. In this case, the information is broadcasted to all neighbors, including the (x, y) coordinates, and the level of s is set to 0. There are three cases in which a node doesn't become a cluster head, but a child node:

1 When $level(s) = -1$, node s receives a broadcast message from its neighbor n, including the (x, y) coordinates of its cluster head h_n. If $||sh_n|| < R$, s chooses h_n as its cluster head. $level(s) := level(n) + 1$ and the distance of s from its cluster head is set to $||sn|| + ||nh_h||$.

2 If s receives a message from neighbor n and $level(s) \neq -1$, node s has already chosen its cluster head. If n is assigned to a different cluster head h_n whose distance from s is in cluster range R and the previously calculated distance to its current cluster head is greater than $||sn|| + ||nh_n||$, then h becomes the new cluster head of s and $level(s) := level(n) + 1$.

3 If s receives a message from neighbor n, $level(s) \neq -1$ and the cluster heads of s and n are the same, it is checked whether the distance of node s to its neighbor n is less than the previously calculated distance. If it is, s will choose n as its parent and set $level(s)$ and the new distance as in the second case.

For finalization, the cluster generation is run a last (seventh) time. Afterwards, each node is either a cluster head or a child node.

In comparison to hierarchical control clustering, DWEHC converges in a constant number of iterations. Moreover, it not only respects the distance between nodes, but also their residual energy. In contrast to previously proposed protocols like LEACH [33], DWEHC doesn't require knowledge about the network size, density or homogeneity or about the number of levels, like HEED [63]. While cluster topologies generated by HEED may not achieve minimum energy consumption in intra-cluster communication, it was shown empirically that the energy savings of DWEHC outperform those or HEED. Also, DWEHC produces more well-balanced clusters and a better distribution of cluster heads, resulting in higher energy savings for inter-cluster communication.

2.1.3 Further Reading. Hierarchical node clustering and DWEHC are only representatives of several distributed clustering algorithms that have been developed for WSNs. The survey article by Abbasi and Younis [1] gives a thorough summary of many additional algorithms. For example, other clustering approaches that have a linear convergence rate are LCA [4], CLUBS [42], RCC [42] and EEHC [5]. Further approaches with a constant number of iterations are, for example, LEACH [33], HEED [63], MOCA [64], EECPL [3] or N-LEACH [56].

2.2 Distributed Clustering of Sensor Measurements

The distributed algorithms described in the previous section cluster sensor nodes. Their purpose is to determine a node topology that allows for an energy-efficient gathering of data, *i.e.* sensor measurements, from the network. As an unsupervised method, clustering can also be used for an exploratory analysis of data, finding groups of similar sensor measurements. Traditional clustering algorithms usually assume all data to be available at a single site, like a base station. Even with an established network topology that allows for energy-efficient communication, due to energy-constraints it is usually not feasible to transfer all available sensor measurements to a single site for clustering. Instead, distributed algorithms need to process data in-network, locally at the sensor nodes, and respect the given limitations of WSNs as much as possible when communicating with other nodes.

Distributed clustering algorithms have been developed in distributed data mining (DDM). These algorithms are often based on the parallel computing paradigm. Running time should be improved by moving data over high-bandwidth connections from a central location to so called compute nodes, and then working on subsets of the data in parallel.

The algorithms usually have full control over where to place the data. Moreover, the cost model for communication only takes into account the time needed for transferral, but not the consumption of energy. In WSNs, however, there is much less control over the data partitioning. For example, application constraints may prohibit certain combinations of sensor placements. Also, the most limiting factor in WSNs is energy, not necessarily time. For this reason, algorithms developed according to the parallel computing paradigm are usually not well-suited for WSNs. In comparison to the large number of algorithms that form clusters of sensor nodes, currently only few algorithms exist that efficiently cluster the sensor measurements themselves.

Generally, clustering algorithms situated in and developed for peer-to-peer networks are good candidates for use in WSNs, especially those that mostly rely on local computations and communication with a limited number of nearest neighbor nodes only. Datta *et al.* [18] have introduced two distributed variants of the k-Means algorithm. The first variant, LSP2P (Local Synchronized-Based P2P) k-Means, is based on a more general local algorithm for mining data streams in distributed systems [58]. It carries out repeated iterations of a modified k-Means at each local node and collects newly calculated centroids and cluster counts only from its immediate neighbors to produce the centroids for the next iteration. Nodes terminate if these new centroids don't differ substantially from the old ones. The algorithm requires no global synchronization and can be extended to a dynamic environment, in which nodes enter and leave the network. Communication costs are shown to be independent of the number of observations to cluster and the total amount of communication is $O(nI(K+L))$, where n is the number of nodes, I the number of iterations, K the number of clusters and L the maximum number of neighbors. It was shown empirically that the algorithm yields similar accuracy as a centralized version of k-Means, however, proving convergence or bounds on the accuracy appears to be a hard problem. The second variant, USP2P (P2P k-Means Clustering Based on Uniform Node Sampling), improves the work by Bandyopadhyay *et al.* [6] and selects s nodes randomly uniformly by a random walk strategy [19] to update centroids in each iteration. For a static network, USP2P provides an accuracy guarantee. Communication costs are upper bounded by $O(Ms\log(n))$, where M denotes the maximum allowed number of iterations by source node, s is the random walk length and n is the number of nodes.

Nowak [44] has introduced DEM, a distributed expectation maximization algorithm for clustering data from a Gaussian mixture distribution, with a particular focus on sensor networks. DEM utilizes an incremen-

tal version of the EM algorithm [43]. It repeatedly cycles through all nodes in a network and performs incremental E- and M-steps at each node, using only locally stored data and summary statistics passed from the previous node. DEM is guaranteed to converge to a local maximum and, as shown empirically, often more rapidly than the standard EM algorithm. Gu [28] proposes to estimate the global sufficient statistics for the M-step by an average consensus filter, diffusing the local sufficient statistics over the entire network by communicating only with neighboring nodes. Thereby, each node gradually gains global information, until the parameters to estimate can be accessed from any node in the network. The local communication between neighbors which is inherently parallel makes it more run-time efficient than DEM which repeatedly cycles over all nodes in the network. Another approach by Kriegel *et al.* [37], the DMBC (Distributed Model-Based Clustering) algorithm, also assumes a Gaussian mixture distribution. It first estimates the number of Gaussian clusters, their parameters (mean and covariance matrix) and their weights at the local nodes, using the standard EM algorithm. Then, the local parameters and weights are transferred to a central site, where similar Gaussians are joined to a compact global distribution. The similarity is measured as the *mutual support* between two clusters C_1, C_2, which in addition to their mean vectors also considers the variance of the clusters. For high dimensional data, DMBC assumes attributes to be independent of each other, resulting in a reduction of the $d \times d$ covariance matrices to d-dimensional variance vectors. For n nodes and a maximum number of local clusters K, the total communication costs are thus bounded by $O(nK)$. It was shown empirically, for varying numbers of clusters and nodes, that the clustering found by DMBC is highly similar to a central clustering, as measured by the Rand Index.

Further algorithms exist, like a distributed version of density-based clustering [34], spectral clustering [51] and solutions specialized on particular applications, like spatial [41] and time series clustering [62].

Only few algorithms developed so far are truly resource-aware and consider, for example, the residual energy of nodes or the CPU utilization explicitly. An exception is ERA-cluster, proposed by Phung *et al.* [46], which is based on the concept of microclusters [2]. It can automatically adapt its sampling rate and the number of examined microclusters based on the current battery, memory and CPU utilization as measured by a resource monitor. EDISKCO [31] solves the k-center clustering problem and can also determine outliers. It works incrementally and only needs a single pass over the input observations, without storing them. The local nodes keep a special heap structure for storing their local k centers and z outliers, sorted according to cluster counts. If a new point doesn't

fit the current clustering, a request for increasing the radius is sent to a coordinator. The coordinator replies with the biggest radius it has received from all other nodes. The local nodes maintain their heap such that the effect of the most l dense clusters which appeared in history solutions is kept, but also such that space is left for establishing new clusters if there is a new trend in the input stream. The coordinator receives the local solutions C_i, radii R_i and radius increase requests from the nodes. It continuously performs the Furthest Points algorithm on the solutions C_i and keeps the largest radius received from all nodes. The base station (server side) rotates the coordinator according to an estimate of the residual energy in each node. EDISKCO determines a $(4 + \varepsilon)$-approximation of the optimal global clustering. Empirically, it was shown that the algorithm outperforms the centralized Global Parallel Guessing algorithm that was proposed by Cormode *et al.* [16], with regard to accuracy as well as energy consumption.

Incorporating energy saving techniques from sensor node clustering into methods for distributed data analysis, like regularly switching the role of the central coordinator, seem to be a fruitful area of future research. This not only concerns the clustering of sensor measurements, but also methods for classification and prediction, like the ones presented in the following section.

3. Classification in Wireless Sensor Networks

Collaborative target classification is an active area of research in the WSN community. US government funded projects through DARPA and Department of Defense (DoD) are interested in a variety of sensor networks applications for modern warfare. One such classic application is multi-vehicle tracking and classification using distributed wireless sensor networks. The goal here is two fold. Since sensors are deployed across the hostile terrain, the first goal is to develop collaborative models which use the data of all sensors and then deploy distributed data mining techniques to build such models using low power consumption and communication overhead. A major advantage of using such collaborative techniques is to bolster the inference of one node using the posterior of the other node. In essence, if one node can validate a hypothesis, then in makes more sense to use it for subsequent inferencing rather than starting from scratch for each node. This forms the second goal of such inferencing technique. Such a collaborative system was developed and deployed by Meesookho *et al.* [40] for identifying and classifying vehicle types from a convoy of vehicles. The paper shows that using confidence boosting, which uses the posterior of one node to

do inference on the next node, the classification accuracy increases by 7%, while the collaborative data driven approach boosts the accuracy by 9%. Finally, the paper shows how collaborative mining techniques can help in identifying and isolating the effects of multiple vehicles which is itself a very hard problem due to signal interference.

A similar approach is discussed in the work by D'Costa and Sayeed [20]. In their work they introduce the concept of collaborative signal processing (CSP). This approach can be used to minimize the amount of information that is passed among the sensor nodes. Two forms of CSP are discussed in the paper: (1) *data fusion*: which exchanges low dimensional feature vectors between the correlated nodes for optimal network performance, and (2) *decision fusion*: which exchanges likelihood values among the independent nodes. The latter one is preferred in many sensor network situations due to its low computational and communication overhead. This paper studies CSP algorithms for single target classification based on multiple acoustic signals measured at different nodes. One of the ways sensor networks can save power is by using a region-based processing instead of all nodes communicating to each other. A manager node is assigned to each region which coordinates the communication among the nodes in each region and also across different regions. In this model, single target classification consists of the following steps: (1) *target detection and classification*: the first step is to use CSP algorithm to detect the region in which the target is, and designate it as the active region, (2) *target localization*: this step is used by the manager nodes to localize the target using the energy detected at each node, (3) *target location prediction*: past estimates are used by the manager nodes to predict future values, and (4)*active location determination*: when the target becomes close to any other region, that region is designated as the new active region and this process is repeated. This paper studies three classifiers – a optimum maximum likelihood classifier, a data averaging classifier that treats all measurements as correlated, and a decision-fusion classifier that treats each observation as independent. Experimental results on DARPA SensIT program data the sub-optimal decision fusion classifier is the most attractive model in the sensor network context.

Researchers have also published several papers on data classification in sensor networks. One such method is the hierarchical decision tree classification technique proposed by Cheng *et al.* [14]. The basic idea of this method is to first construct a spanning tree encompassing all the nodes in the system. Construction of the classifier (decision tree in this context) begins with the leaves nodes of the spanning tree first building a decision tree C_i with only its local data and sending these upstream

to the parent nodes in the spanning tree. The parent nodes of the spanning tree then builds a new classifier by combining all the classifiers it has received from its children and by subsampling a portion of the dataset with same proportion of negative and positive examples. These intermediate nodes then send the classifiers again upstream and the base station builds a single classifier which represent all the data over all the nodes. The paper discusses the fact that a short but wide spanning tree increases the communication cost of sending the classifiers to the next node but reduces the overall accuracy due to smaller number of hops, while a tall but narrow tree suffers from the opposite effect. Finally, the paper presents extensive experimental results on simulated wireless testbed to show that this method offers better accuracy and energy consumption compared to a baseline ensemble method in which meta classifiers are learned independently at each node and then (majority) voting is applied during test phase.

The above algorithm suffers from one major drawback — it requires synchronization in every time step and hence can be expensive to deploy for the next generation of large sensor networks. In a recent paper, Bhaduri *et al.* [9] have proposed a decision tree learning algorithm which can build the same tree on all the nodes in an asynchronous fashion. The main building block of the algorithm is the scalable distributed majority voting protocol first discussed in the paper by Wolff and Schuster [59]. Given a pair of real numbers a_i and b_i at each node, this algorithm decides if $\sum_i a_i > \sum_i b_i$ in a very communication efficient fashion, without needing a node to exchange messages even if a_i and b_i are changing. Based on this protocol, first, the authors show that comparison of two features can be accomplished by concurrently running 4 majority votes. The next step is to choose top 1-out-of-k attributes and this can be easily accomplished by running the previous comparison per attribute pair. Finally, the tree can be built asynchronously by performing this 1 out of k comparison for each level of the tree. First of all, this algorithm is guaranteed to converge to the globally correct solution on convergence. Extensive experimental results also show that the algorithm is communication efficient, even when the data is changing.

Probabilistic gossip based protocols have been used extensively for many WSN algorithms due to their simplicity in implementation and asymptotically guaranteed convergence. Distributed consensus algorithms such as averaging, summation, max/min etc. can be efficiently computed using gossip protocols in which a node randomly chooses another node and exchanges information with it. This process continues for some iterations whereby it can be shown that the error reduces exponentially at each iteration. Using such a protocol, Chiuso *et al.* [15]

have proposed an algorithm for distributed classification and estimation in wireless sensor networks. The data is modeled as

$$y_i = \theta + T_i + \nu_i$$

where y_i's are the measurements at each sensor node, $\theta \in \mathbb{R}$ is the common unknown parameter, $T_i \in \{0, 1\}$ are the unknown discrete terms which denotes the class label of each node, and ν_i's are zero mean iid Gaussian random variables with finite variance. The goal of each node is to estimate θ and T_i. Due to the existence of θ, the final estimate would require a consensus algorithm over all nodes. The paper develops a maximum likelihood estimator to estimate the unknown parameter and infer the class labels in the distributed setting using a gossip based protocol. The paper further proposes an EM algorithm for the case in which the T_i's are assumed to be iid Bernoulli trials. Experimental results show that the proposed methods have similar convergence rates compared to existing methods but stronger robustness in various situations, for instance when the offset of the misbehaving sensors is not known, or in the presence of outliers.

Further reading: There are a number of other papers in these areas which we point out here. Sun and Qi [54] discuss the fact that there exist a particular set of features and a particular classifier which has the best performance, in terms of highest accuracy with the least number of features used. The authors discuss a method of dynamic target classification in which an optimal set of features and classifiers are determined based on some minimal value of cost function. Experimental results show that this approach can significantly reduce the computational time and also achieve better classification accuracy.

Eyal *et al.* [26] present an asynchronous algorithm for distributed data classification over arbitrary connected networks. They present a generic algorithm converges for any connected topology, data and class distribution. The paper presents examples of two specific instantiations of the generic algorithm: (1) a distance based classification scenario akin to the famous k-means clustering, and (2) a gaussian mixture model data distribution with expectation maximization for learning latent factors.

Duarte and Hu [22] discuss the application of vehicle classification in sensor networks. Each sensor in the WSN is equipped with a microphone or a geophone. Upon detection of the presence of a vehicle in the vicinity of the sensor, the on-board processor first extracts features in the frequency domain using FFT. The next step is to use a local classifier at each node to generate a preliminary hypothesis about the observation using only the data present at that node. The authors have experimented with 3 classifiers – a k-nn based classifier, a maximum like-

lihood classifier, and an SVM classifier. The local decision, together with the estimated probability of being a correct decision is transmitted to a local fusion center rather than sending the raw data. The fusion center can then use maximum aposterior (MAP) estimate to compute the final decision on the classification of the observation. Extensive experimental results show that the MAP estimate with the nearest neighbor as the local classifiers works well in vehicle classification.

Some other important work in this area include the distributed target classification work by Brooks *et al.* [11], Gu *et al.* [29], and Kotecha *et al.* [36].

Another area similar to distributed classification in WSN is distributed event detection. The main goal is to detect frequent event patterns based on some data mining models while minimizing the need for communication all the data from all the nodes to the sink. One such method is the technique based on frequent itemset mining by Römer [49][50]. First local association rules are learned at each node and then these rules along with the support and confidence are sent to the sink. Experimental results demonstrate that this method is efficient and detects correct frequent events. Wittenburg *et al.* [57] present a method for distributed event detection. Their method consists of sampling the data in the network, feature selection and then learning a model at each node.

Tavakoli *et al.* [55] consider a scenario in which targets are tracked using an undersea acoustic sensor network. The sensor nodes report their local classification result to a cluster head which then in turn performs an evaluation of the data and may report the outcome to a base station. As a confidence interval, the method considers the accuracy of these past reports.

The system proposed by Yang *et al.* [61] is aimed at recognizing human motions. It is a wearable sensor system consisting of eight sensor nodes attached to the body of a person who may perform one out of twelve actions. Accelerometer and gyroscope are used to detect the motions and then features are extracted and classified at each node to detect events. If a local classification is promising, the data of all nodes is transmitted to the base station and classified once again. The classification process identifies an action by matching the linear representation of the extracted feature vector to one of several subspaces, each of which corresponds to one type of action.

4. Outlier Detection in WSN

Outlier detection is one of the most critical tasks performed in WSNs due to their ability to monitor hostile environments. In general, the

term outlier has many definitions, but the core problem is to detect points or observations from the dataset that are different compared to the others. In WSN environments, this translates to finding points which are compared to all the points that are sensed by the sensors [12]. There are many applications of anomaly detection in WSNs. Here we briefly list some of these here:

- Environmental monitoring: sensors are deployed in harsh environments to monitor events that occur in natural environments
- Habitat monitoring of species or animals for conversation purposes or for understanding their migration patterns
- Health and medical monitoring tasks in which the goal is to use different kinds of non-intrusive wearable sensors (*e.g.* acoustic, temperature, SO_2, pressure) to analyze the health of humans
- Industrial monitoring: sensors are used to sense the health or condition of industrial processes
- Target tracking and surveillance, sensors are embedded in moving targets to track them in real-time

In all of these applications, there is a real need for anomaly detection for further analysis of abnormal observations. The task of outlier detection in WSN is extremely difficult mainly because of these reasons [27]: (1) resource constraints, (2) high communication and computation cost, (3) distributed streaming data, (4) asynchronous computation model, (5) large scale deployment, and (6) dynamic network topology, to name a few. In the remainder of this section we discuss several techniques for outlier detection in WSNs following the taxonomy given in Chandola *et al.* [12] and Zhang *et al.* [66].

4.1 Statistical approaches

In statistical approaches, the task is to model the probability distribution of the data using parametric or non-parametric approaches and then tag as outliers those data points which do not fit the modeled distribution.

Wu *et al.* [60] present two local techniques for identification of outlying sensors. These techniques employ the spatial correlation of the readings existing among neighboring sensor nodes to detect bad sensors. Each node computes the distance between its own reading and the median reading of its neighboring sensors. A node is considered as an outlying node if, the absolute value of this distance is sufficiently large

compared to a pre-selected threshold. Accuracy of these outlier detection techniques is not relatively high due to the fact that they ignore the temporal correlation of sensor readings.

Battencourt *et al.* [8] present a technique for outlier detection in WSNs for ecosystem monitoring applications. The method exploits spatio-temporal data distribution to find outliers. The basic idea is to compare the measurement of one sensor with those in the spatial vicinity and also with its measurements back in time. Then, if the deviation of these values are greater than a user defined threshold (based on a statistical significance test), a sensor detects an outlier. The obvious drawback of this method is the choice of the outlier.

In a set of different approaches, researchers have proposed non-parametric methods for anomaly detection. Two such approaches are histogram computation and kernel density estimation (KDE). Sheng *et al.* [52] present a histogram-based technique to identify global outliers in WSN. Instead of transmitting raw data back to the base station for processing, this technique first builds data histogram at local nodes and the ships these statistics to the base station (sink). The sink uses this histogram information to extract data distribution from the network and filters out the non-outliers. The identification of outliers is achieved by a fixed threshold distance or the rank among all outliers. One of the major drawbacks of this technique is the ability to process only one dimensional data. Subramaniam *et al.* [53] and Palpanas *et al.* [45] present techniques for outlier detection using kernel density estimation. Instead of comparing all the raw observations, the technique fits kernel densities at each of the observation points which considerably smooths the values. Then user defined thresholds are applied in order to identify outliers. Experimental results show that these techniques achieve high accuracy in terms of estimating data distribution and high detection rate while consuming low memory usage and message transmission.

4.2 Nearest neighbor based approaches

Nearest neighbor approaches use distance to other points to compute an outlier. One of the widely used definitions, based on the original idea of Knorr *et al.* [35], is that outliers are those points which are *very far* from its nearest neighbors. Many variants of this definition have been proposed based on the definition of distance and the threshold for choosing how "far". One practical definition uses Euclidean distance and a user defined threshold or the number of desired outliers.

Such a definition has been used by Branch *et al.* [10] to find global outliers in WSNs. The basic idea is to use a set of local rules by which a

node determines outliers in its local dataset, and then broadcasts them to other nodes for validation. The neighboring nodes repeat the procedure until all of the sensor nodes in the network eventually agree on the global outliers. This technique can be flexible with respect to multiple existing distance-based outlier detection techniques. It has two major advantages: (1) the outliers found this method are provably the same that a centralized algorithm would find, and (2) the algorithm can easily adopt to data and network changes. Because of these two advantages, the technique is greatly suitable for WSNs. However, one drawback of this method is that it requires a node to broadcast all the outliers to all the other nodes for validation.

Zhang *et al.* [65] propose a distance-based technique to identify n global outliers in continuous query processing applications of sensor networks. To overcome the broadcast issue of Branch *et al.* [10], [65] adopts the structure of aggregation tree that do not require broadcasting of each node in the network. Each node in the tree transmits some useful data to its parent after collecting all the data sent from its children. The sink then approximates the top n global outliers and sends these outliers to all the nodes in the network for verification. If any node disagrees on the global results, it will send extra data to the sink again for outlier detection. This procedure is repeated until all the nodes in the network agree on the global results calculated by the sink. A major drawback of this technique is that it requires a tree topology to be overlaid on top of the network and hence not suitable for any topology types.

4.3 Classification based approaches

Given examples of two kinds, an outlier detection problem can be transformed to a classification problem. This trick has been widely explored in the data mining community and Chandola *et al.* [12] presents a good overview on this topic. Even in the area of WSN, the classification techniques that we have presented in Section 3 can be applied for outlier detection in WSNs. One such example is the one-class support vector machines algorithm that can learn a non-linear hyper surface via the kernel trick. Rajasegarar *et al.* [48] use this model for outlier detection. In the first phase of this technique, a local model is learned at each node and then points which are outside this model are sent to the sink node along with the model. These local outliers are then validated and the global set is determined.

Two other approaches have been explored for classification in WSNs. Bayesian approaches such as naive bayes, dynamic bayes and bayesian belief propagation models have been used by Elnahrawy and Nath [23]

and others. Finally, spectral clustering methods using eigen decomposition techniques have been proposed by Chatzigiannakis *et al.* [13].

5. Conclusions

Distributed data mining will continue to play an important role in analysis of data in modern sensor networks. Since computation is sensor networks is greatly constrained by the various challenges facing a modern WSN, a need breed of data mining algorithms need to be developed which can co-analyze the data sensed by all the sensors by paying careful attention to computation, communication and any other constraints. To circumvent this problem, several algorithms have been proposed that can effectively handle the harsh environments of WSNs. In this chapter we have discussed three such topics related to data mining in sensor networks, viz., clustering, classification and outlier detection. Of course, we have only been able to scratch the surface of this vast area of research. With WSNs being deployed in many realms of life for monitoring purposes, distributed data mining is likely to play a critical role and thus offers plenty of opportunities for both novel algorithm development and data analysis.

Acknowledgements

The second author acknowledges funding by the DFG, Collaborative Research Center SFB 876, project B3.

References

[1] A.A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Comput. Commun.*, 30(14-15):2826–2841, Oct. 2007.

[2] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for clustering evolving data streams. In *Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB)*, pages 81–92, 2003.

[3] F. Bajaber and I. Awan. Energy efficient clustering protocol to enhance lifetime of wireless sensor network. *Journal of Ambient Intelligence and Humanized Computing*, 1:239–248, 2010.

[4] D. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Trans. on Communications*, 29(11):1694–1701, Nov. 1981.

[5] S. Bandyopadhyay and E.J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proc. of*

the 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM), volume 3, pages 1713–1723, Apr. 2003.

[6] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176(14):1952–1985, 2006.

[7] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *Proc. of the 20th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1028–1037, 2001.

[8] L.M.A. Bettencourt, A.A. Hagberg, and L.B. Larkey. Separating the wheat from the chaff: practical anomaly detection schemes in ecological applications of distributed sensor networks. In *Proc. of the 3rd IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, pages 223–239, 2007.

[9] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta. Distributed decision tree induction in P2P systems. *Stat. Anal. and Data Mining*, 1(2):85–103, 2008.

[10] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *Proc. of the 26th IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, page 51, 2006.

[11] R.R. Brooks, P. Ramanathan, and A.M. Sayeed. Distributed target classification and tracking in sensor networks. In *Proc. of the IEEE*, pages 1163–1171, 2003.

[12] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, Jul. 2009.

[13] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris. Hierarchical anomaly detection in distributed large-scale sensor networks. In *Proc. of the 11th IEEE Symp. on Computers and Communications*, pages 761–767, 2006.

[14] Xu Cheng, Ji Xu, Jian Pei, and Jiangchuan Liu. Hierarchical distributed data classification in wireless sensor networks. *Comput. Commun.*, 33(12):1404–1413, 2010.

[15] A. Chiuso, F. Fagnani, L. Schenato, and S. Zampieri. Gossip algorithms for simultaneous distributed estimation and classification in sensor networks. *J. Sel. Topics Signal Processing*, 5(4):691–706, 2011.

[16] G. Cormode, S. Muthukrishnan, and Wei Z. Conquering the divide: Continuous clustering of distributed data streams. In *Proc. of the*

23rd IEEE Int. Conf. on Data Eng. (ICDE), pages 1036–1045, Apr. 2007.

[17] W. Dargie and C. Poellabauer. *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons, 2010.

[18] S. Datta, C. Giannella, and H. Kargupta. Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Trans. on Knowl. and Data Eng.*, 21(10):1372–1388, Oct. 2009.

[19] S. Datta and H. Kargupta. Uniform data sampling from a peer-to-peer network. In *Proc. of the 27th IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, page 50, 2007.

[20] A. D'Costa and A.M. Sayeed. Collaborative signal processing for distributed classification in sensor networks. In *Information Processing in Sensor Networks*, volume 2634 of *Lecture Notes in Computer Science*, pages 558–558. Springer, 2003.

[21] P. Ding, J. Holliday, and A. Celik. Distributed energy-efficient hierarchical clustering for wireless sensor networks. In *Proc. of the 1st IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, pages 322–339. Springer, 2005.

[22] M.F. Duarte and Y.H. Hu. Vehicle classification in distributed sensor networks. *J. Parallel Distrib. Comput.*, 64(7):826–838, Jul. 2004.

[23] E. Elnahrawy and B. Nath. Context-aware sensors. In *Proc. of the 1st European Conf. on Wireless Sensor Networks (EWSN)*, pages 77–93, 2004.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd Int. Conf. on Knowl. Discovery and Data Mining*, pages 226–231, 1996.

[25] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proc. of the 27th IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[26] I. Eyal, I. Keidar, and R. Rom. Distributed data classification in sensor networks. In *Proc. of the 29th ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pages 151–160, 2010.

[27] M.M. Gaber, R.R. Vatsavai, O.A. Omitaomu, J. Gama, N.V. Chawla, and A.R Ganguly, editors. *Knowledge Discovery from Sensor Data*. Springer, 2008.

[28] D. Gu. Distributed EM algorithm for gaussian mixtures in sensor networks. *IEEE Trans. on Neural Networks*, 19(7):1154–1166, Jul. 2008.

[29] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J.A. Stankovic, T.F. Abdelzaher, and B.H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *Proc. of the 3rd Int. Conf. on Embedded networked sensor systems (SenSys)*, pages 205–217, 2005.

[30] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2005.

[31] M. Hassani, E. Müller, and T. Seidl. EDISKCO: Energy efficient distributed in-sensor-network k-center clustering with outliers. In *Proc. of the 3rd Int. Workshop on Knowl. Discovery from Sensor Data (SensorKDD)*, pages 39–48, 2009.

[32] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2nd edition, 2009.

[33] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. on Wireless Communications*, 1(4):660–670, Oct. 2002.

[34] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Scalable density-based distributed clustering. In *Proc. of the 8th European Conf. on Principles and Practice of Knowl. Discovery in Databases (OKDD)*, pages 231–244. Springer, 2004.

[35] E.M. Knorr, R.T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, Feb. 2000.

[36] J.H. Kotecha, V. Ramachandran, and A.M. Sayeed. Distributed multitarget classification in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):703–713, 2005.

[37] H.-P. Kriegel, P. Kröger, A. Pryakhin, and M. Schubert. Effective and efficient distributed model-based clustering. In *Proc. of the 5th IEEE Int. Conf. on Data Mining (ICDM)*, pages 258–265, 2005.

[38] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1:1–1:58, Mar. 2009.

[39] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability*, pages 281–297, 1967.

[40] C. Meesookho, S. Narayanan, and C.S. Raghavendra. Collaborative classification applications in sensor networks. In *In Proc. of the*

2nd Sensor Array And Multichannel Signal Processing Workshop (SAM), pages 370–374, 2002.

[41] A. Meka and A.K. Singh. Distributed spatial clustering in sensor networks. In *Proc. of the 10th Int. Conf. on Advances in Database Technology (EDBT)*, pages 980–1000. Springer, 2006.

[42] R. Nagpal and D. Coore. An algorithm for group formation in an amorphous computer. In *Proc. of the 10th Int. Conf. on Parallel and Distributed Computing Systems (PDCS)*, 1998.

[43] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in graphical models*, pages 355–368. MIT Press, 1999.

[44] R.D. Nowak. Distributed EM algorithms for density estimation and clustering in sensor networks. *IEEE Trans. on Signal Processing*, 51(8):2245–2253, Aug. 2003.

[45] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Rec.*, 32(4):77–82, Dec. 2003.

[46] N.D. Phung, M.M. Gaber, and U. Rohm. Resource-aware online data mining in wireless sensor networks. In *IEEE Symp. on Comput. Intelligence and Data Mining (CIDM)*, pages 139–146, Apr. 2007.

[47] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, May 2000.

[48] S. Rajasegarar, C. Leckie, M. Palaniswami, and J.C. Bezdek. Quarter sphere based distributed anomaly detection in wireless sensor networks. In *Proc. of the IEEE Int. Conf. on Communications (ICC)*, pages 3864–3869, 2007.

[49] K. Römer. Distributed mining of spatio-temporal event patterns in sensor networks. In *Euro-American Workshop on Middleware for Sensor Networks in conjunction with DCOSS 2006*, pages 103–116, June 2006.

[50] K. Römer. Discovery of frequent distributed event patterns in sensor networks. In *Proc. of the 5th European Conf. on Wireless sensor networks*, pages 106–124, 2008.

[51] T. Sahai, A. Speranzon, and A. Banaszuk. Hearing the clusters of a graph: A distributed algorithm. *Automatica*, 48(1):15–24, Jan. 2012.

[52] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *Proc. of the 8th ACM Int. Symp. on Mobile ad hoc networking and computing*, pages 219–228, 2007.

[53] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proc. of the 32nd Int. Conf. on Very Large Data Bases (VLDB)*, pages 187–198, 2006.

[54] Y. Sun and H. Qi. Dynamic target classification in wireless sensor networks. In *Proc. of the 19th Int. Conf. on Pattern Recognition (ICPR)*, pages 1–4, Dec. 2008.

[55] A. Tavakoli, J. Zhang, and S. H. Son. Group-based event detection in undersea sensor networks. In *Proc. of 2nd Intl. Workshop on Networked Sensing*.

[56] R.K. Tripathi, Y.N. Singh, and N.K. Verma. N-leach, a balanced cost cluster-heads selection algorithm for wireless sensor network. In *National Conference on Communications (NCC)*, pages 1–5, Feb. 2012.

[57] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller. A system for distributed event detection in wireless sensor networks. In *Proc. of the 9th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks*, pages 94–104, 2010.

[58] R. Wolff, K. Bhaduri, and H. Kargupta. A generic local algorithm for mining data streams in large distributed systems. *IEEE Trans. on Knowl. and Data Eng.*, 21(4):465–478, Apr. 2009.

[59] R. Wolff and A. Schuster. Association rule mining in peer-to-peer systems. *IEEE Trans. on Systems, Man and Cybernetics - Part B*, 34(6):2426–2438, Dec. 2004.

[60] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng. Localized outlying and boundary data detection in sensor networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1145–1157, Aug. 2007.

[61] A. Yang, R. Jafari, S. Sastry, and R. Bajcsy. Distributed recognition of human actions using wearable motion sensor networks. *J. Ambient Intell. Smart Environ.*, 1(2):103–115, 2009.

[62] J. Yin and M.M. Gaber. Clustering distributed time series in sensor networks. In *Proc. of the 8th IEEE Int. Conf. on Data Mining (ICDM)*, pages 678–687, 2008.

[63] O. Younis and S. Fahmy. HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Trans. on Mobile Computing*, 3(4):366–379, 2004.

[64] A. Youssef, M. Younis, M. Youssef, and A. Agrawala. Distributed formation of overlapping multi-hop clusters in wireless sensor networks. In *IEEE Global Telecommunications Conf. (GLOBECOM)*, pages 1–6, Dec. 2006.

[65] K. Zhang, S. Shi, H. Gao, and J. Li. Unsupervised outlier detection in sensor networks using aggregation tree. In *Proc. of the 3rd Int. Conf. on Advanced Data Mining and Applications (ADMA)*, pages 158–169, 2007.

[66] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Commun. Surveys Tuts.*, 12(2):159–170, Apr. 2010.

Chapter 9

SOCIAL SENSING

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY
charu@us.ibm.com

Tarek Abdelzaher
University of Illinois at Urbana Champaign
Urbana, IL
zaher@cs.uiuc.edu

Abstract A number of sensor applications in recent years collect data which can be directly associated with human interactions. Some examples of such applications include GPS applications on mobile devices, accelerometers, or location sensors designed to track human and vehicular traffic. Such data lends itself to a variety of rich applications in which one can use the sensor data in order to model the underlying relationships and interactions. This requires the development of trajectory mining techniques, which can mine the GPS data for interesting social patterns. It also leads to a number of challenges, since such data may often be private, and it is important to be able to perform the mining process without violating the privacy of the users. Given the open nature of the information contributed by users in social sensing applications, this also leads to issues of trust in making inferences from the underlying data. In this chapter, we provide a broad survey of the work in this important and rapidly emerging field. We also discuss the key problems which arise in the context of this important field and the corresponding solutions.

Keywords: Sensor Networks, Social Sensors, Cyber-physical Networks

1. Introduction

The proliferation of numerous online social networks such as *Facebook*, *LinkedIn* and *Google+* has lead to an increased awareness of the power of incorporating social elements into a variety of data-centric applications. Such networks are typically *data rich*, and contain heterogeneous data along with linkage stricture, which can be mined for a variety of purposes [39, 98, 108]. In particular, it has been observed that the use of a combination of social structure and different kinds of data can be a very powerful tool for mining purposes [136, 175, 182]. A natural way to enhance the power of such social applications is to embed sensors within such platforms in order to continuously collect large amounts of data for prediction and monitoring applications. This has lead to the creation of numerous social sensing systems such as *Biketastic* [142], *BikeNet* [55], *CarTel* [88] and *Pier* [148], which use social sensors for a variety of transportation and personal applications. The fusion of mobile, social, and sensor data is now increasingly being seen as a tool to fully enable context-aware computing [20].

A number of recent hardware platforms have extended the data-centric capabilities of social networks, by providing the ability to embed sensor data collection directly into the social network. Therefore, it is natural to explore whether sensor data processing can be tightly integrated with social network construction and analysis. For example, methods such a crowd-sourcing are a natural approach for improving the accuracy of many socially-aware search applications [168]. Some of the afore-mentioned data types on a conventional social network are static and change slowly over time. On the other hand, sensors collect vast amounts of data which need to be stored and processed in real time. There are a couple of important drivers for integrating sensor and social networks:

- One driver for integrating sensors and social networks is to allow the actors in the social network to both publish their data and subscribe to each other's data either directly, or indirectly after discovery of useful information from such data. The idea is that such collaborative sharing on a social network can increase real-time awareness of different users about each other, and provide unprecedented information and understanding about global behavior of different actors in the social network. The vision of integrating sensor processing with the real world was first proposed in [177].

- A second driver for integrating sensors and social networks is to provide a better understanding and measurement of the aggre-

gate behavior of self-selected communities or the external environment in which these communities function. Examples may include understanding traffic conditions in a city, understanding environmental pollution levels, or measuring obesity trends. Sensors in the possession of large numbers of individuals enable exploiting the crowd for massively distributed data collection and processing. Recent literature reports on several efforts that exploit individuals for data collection and processing purposes such as collection of vehicular GPS trajectories as a way for developing street maps [78], collectively locating items of interest using cell-phone reports, such as mapping speed traps using the Trapster application [190], use of massive human input to translate documents [145], and the development of protein folding games that use competition among players to implement the equivalent of global optimization algorithms [21].

The above trends are enabled by the emergence of large-scale data collection opportunities, brought about by the proliferation of sensing devices of every-day use such as cell-phones, piedometers, smart energy meters, fuel consumption sensors (standardized in modern vehicles), and GPS navigators. The proliferation of many sensors in the possession of the common individual creates an unprecedented potential for building services that leverage massive amounts data collected from willing participants, or involving such participants as elements of distributed computing applications. Social networks, in a sensor-rich world, have become inherently multi-modal data sources, because of the richness of the data collection process in the context of the network structure. In recent years, sensor data collection techniques and services have been integrated into many kinds of social networks. These services have caused a computational paradigm shift, known as *crowd-sourcing* [23, 47], referring to the involvement of the general population in data collection and processing. Crowd-sourcing, arguably pioneered by programs such as SETI, has become remarkably successful recently due to increased networking, mobile connectivity and geo-tagging [1]. We note that the phenomenon of crowd-sourcing is not exclusive to sensor data, but is also applied to other tagging and annotation processes, in which the knowledge is sourced from a social network of users. A classic example of a crowd-sourcing application is the *Amazon Mechanical Turk* [192], which allows users to submit data records for annotation at the payment of a fee for annotation purposes. Thus, the *Amazon Mechanical Turk* serves as an intermediary for crowd-sourcing of annotations for data records.

In the case of *social sensing* which is also often referred to as *people-centric sensing* [6, 26, 123] or *participatory sensing* [24], this crowd-

sourcing is generally achieved through sensors which are closely attached to humans, either in wearable form, or in their mobile phones. Some examples of integration of social and sensor networks are as follows:

- A variety of applications can be created to collect real time information from large groups of individuals in order to harness the *wisdom of crowds* in a variety of decision processes. For example, the *Google Latitude* application [184] collects mobile position data of uses, and uses this in order to detect the proximity of users with their friends. This can lead to significant events of interest. For example, proximity alerts may be triggered when two linked users are within geographical proximity of one another. This may itself trigger changes in the user-behavior patterns, and therefore the corresponding sensor values. This is generally true of many applications, the data on one sensor can influence data in the other sensors. Numerous other GPS-enabled applications such as *City sense*, *Macrosense*, and *Wikitude* [185, 195, 191] serve as gps-based social aggregators for making a variety of personalized recommendations. The approach has even been used for real-time grocery bargain hunting with the *LiveCompare* system [46].
- *Vehicle Tracking Applications:* A number of real-time automotive tracking applications determine the important points of congestion in the city by pooling GPS data from the vehicles in the city. This can be used by other drivers in order to avoid points of congestion in the city. In many applications, such objects may have implicit links among them. For example, in a military application, the different vehicles may have links depending upon their unit membership or other related data. Two classic examples of vehicular applications in the context of participatory sensing are the *CarTel* [88] and *GreenGPS* [64] systems.
- *Trajectory Tracking:* In its most general interpretation, an actor in a social network need not necessary be a person, but can be any living entity such as an animal. Recently, animal tracking data is collected with the use of radio-frequency identifiers. A number of social links may exist between the different animals such as group membership, or family membership. It is extremely useful to utilize the sensor information in order to predict linkage information and vice-versa. A recent project called *MoveBank* [186] has made tremendous advances in collecting such data sets. We note that a similar approach may be used for commercial product-tracking applications, though social networking applications are generally relevant to living entities, which are most typically people.

- *Applications to Healthcare:* In recent years, numerous medical sensor devices can be used in order to track the personal health of individuals, or make other predictions about their lifestyle [41, 65, 84, 119, 121, 122, 150]. This can be used for emergency response, long term predictions about diseases such as dementia, or other life style influence analysis of factors such as eating habits and obesity.

Social sensing applications provide numerous research challenges from the perspective of analysis. We list some of these challenges below:

- Since the collected data typically contains sensitive personal data (eg. location data), it is extremely important to use privacy-sensitive techniques [61, 133] in order to perform the analysis. A recent technique called *PoolView* [61] designs privacy-sensitive techniques for collecting and using mobile sensor data.

- Sensors, whether wearable or embedded in mobile devices, are typically operated with the use of batteries, which have limited battery life. Certain kinds of sensor data collection can drain the battery life more quickly than others (eg. GPS vs. cell tower/WiFi location tracking in a mobile phone). Therefore, it is critical to design the applications with a careful understanding of the underlying tradeoffs, so that the battery life is maximized without significantly compromising the goals of the application.

- The volume of data collected can be very large. For example, in a mobile application, one may track the location information of millions of users simultaneously. Therefore, it is useful to be able to design techniques which can compress and efficiently process the large amounts of collected data.

- Since the data are often collected through sensors which are error-prone, or may be input by individuals without any verification, this leads to numerous challenges about the *trustworthiness* of the data collected. Furthermore, the goals of privacy and trust tend to be at odds with one another, because most privacy-preservation schemes reduce the fidelity of the data, whereas trust is based on high fidelity of the data.

- Many of the applications require *dynamic* and *real time* responses. For example, applications which trigger alerts are typically time-sensitive and the responses may be real-time. The real-time aspects of such applications may create significant challenges, considering the large number of sensors which are tracked at a given time.

This chapter is organized as follows. Section 2 briefly discusses some key technological advances which have occurred in recent years, which have enabled the design of such dynamic and embedded applications. Section 3 discusses a broad overview of the key system design questions which arise in these different contexts. One of the important issues discussed in this section is privacy, which is discussed in even greater detail in a later section. Section 4 discusses some important privacy issues which arise in the context of social networks with embedded sensors. Section 5 discusses the trust-worthiness issues which arise in such crowd-sourcing systems. Section 6 introduces techniques for social network modeling from dynamic links which are naturally created by the sensor-based scenario. Since such dynamic modeling often requires trajectory mining techniques, we present methods for trajectory mining in section 7. Section 8 introduces some of the key applications associated with social sensing. Section 9 discusses the conclusions and research directions.

2. Technological Enablers of Social Sensing

A number of recent technological advances in hardware and software have enabled the integration of sensors and social networks. One such key technological advance is the development is small mobile sensors which can collect a variety of user-specific information such as audio or video. Many of the applications discussed are based on *user-location*. Such location can easily be computed with the use of mobile GPS-enabled devices. For example, most of the recent smart-phones typically have such GPS technology embedded inside them. Some examples of such mobile sensor devices may be found in [117, 100].

Sensors typically collect large amounts of data, which must be continuously stored and processed. Furthermore, since the number of users in a social network can be very large, this leads to natural scalability challenges for the storage and processing of the underlying streams. For example, many naive solutions such as the centralized storage and processing of the raw streams are not very practical, because of the large number of streams which are continuously received. In order to deal with this issue, a number of recent hardware and software advances have turned out to be very useful.

- *Development of Miniaturized Sensor Technology:* The development of miniaturized (wearable) sensors and batteries have allowed their use and deployment in a number of different social settings. For example, the development of miniaturized sensors, which can be embedded within individual attire can be helpful in a wide vari-

ety of scenarios [42, 100, 63, 33, 34]. A classic example is the spec mote, which is an extremely small sensor device that can be embedded in the clothing of a user, while remaining quite unobtrusive.

- *Advancement of smartphone technology:* In recent years, there has been considerable advancement in smartphone technology, which are now fairly sophisticated devices containing a wide array of sensors such as GPS, compass, accelerometers, bluetooth capabilities etc. In addition, these are *convergent devices*, with considerable computational capabilities, internet connectivity, and different modes of user interaction and content upload, such as social tweets, ability to record pictures and videos etc. All of these capabilities create a rich content-based and sensing environment for a wide variety of applications.

- *Increased Bandwidth:* Since sensor transmission typically requires large wireless bandwidth, especially when the data is in the form of audio or video streams, it is critical to be able to transmit large amounts of data in real time. The increases in available bandwidth in recent years, have made such real time applications a reality.

- *Increased Storage:* In spite of the recently designed techniques for compressing the data, the storage challenges for stream processing continue to be a challenge. Recent years have seen tremendous advances in hardware, which allow much greater storage, than was previously possible.

- *Development of Fast Stream Processing Platforms:* A number of fast stream processing platforms, such as the IBM System S platform [187] have been developed in recent years, which are capable of storing and processing large volumes of streams in real time. This is a very useful capability from the perspective of typical cyber-physical applications which need a high level of scalability for real-time processing.

- *Development of Stream Synopsis Algorithms and Software:* Since the volume of the data collected is very large, it often cannot be collected explicitly. This leads to the need for designing algorithms and methods for stream synopsis construction [7]. A detailed discussion of a variety of methods (such as sketches, wavelets and histograms) which are used for stream synopsis construction and analysis is provided in [7].

The sensing abilities of miniaturized devices and smartphones have also increased considerably in recent years. For example, the one of the earliest systems, which is referred to as a *sociometer* [33, 34], a small wearable device is constructed, which can detect people nearby, provide motion information and accelerometers, and also has microphones for detection of speech information. In addition, the device has the flexibility to allow for the addition of other kinds of sensors such as GPS sensors and light sensors. These sensors can be used in order to detect implicit links between people, and the corresponding community behavior. The aim of collecting a large number of such interactive behaviors is to be able to effectively model interactions, between different users, and then model the dynamics of the interaction with the use of the collected information.

Since the work in [33], much of these sensing capabilities are now available in commodity hardware such as mobile phones. For example, the *Virtual Compass* system [18] uses the sensors available in mobile phones in order to sense the interactions between different actors. Virtual Compass is a peer-based relative positioning system that uses multiple radios to detect nearby mobile devices and places them in a two-dimensional plane. It uses different kinds of scanning and out-of-band coordination to explore tradeoffs between energy consumption, and the latency in detecting movement. Methods are designed for using different kinds of sensor signals in *Virtual Compass* in order to reduce the energy footprint. More details may be found in [18].

3. Data Collection, Architectural and System Design Challenges

The aforementioned monitoring and social computing opportunities present a need for a new architecture that encourages data sharing and efficiently utilizes data contributed by users. The architecture should allow individuals, organizations, research institutions, and policy makers to deploy applications that monitor, investigate, or clarify aspects of *socio-physical* phenomena; processes that interact with the physical world, whose state depends on the behavior of humans in the loop.

An architecture for social data collection should facilitate distillation of concise actionable information from significant amounts of raw data contributed by a variety of sources, to inform high-level user decisions. Such an architecture would typically consist of components that support (i) privacy-preserving sensor data collection, (ii) data model construction, and (iii) real-time decision services. (iv) effective methods for recruitment, and (v) energy efficient design. For example, in an ap-

plication that helps drivers improve their vehicular fuel-efficiency, data collection might involve upload of fuel consumption data and context from the vehicle's on-board diagnostics (OBD-II) interface and related sensors; a model might relate the total fuel consumption for a vehicle on a road segment as a function of readily available parameters (such as average road speed, degree of congestion, incline, and vehicle weight); the decision support service might provide navigation assistance to find the most fuel-efficient route to a given destination (as opposed to a fastest or shortest route). Of course, none of these can be effectively implemented without energy-efficient data collection and participant recruitment. Below, we elaborate on the above functions.

3.1 Privacy-Preserving Data Collection

In a grassroots application that is not managed by a globally trusted authority, an interesting challenge becomes ensuring the privacy of data shared. Anonymity is not a sufficient solution because the data themselves (such as GPS traces) may reveal the identity of the owner even if shared anonymously. One interesting direction is to allow individuals to "lie" about their data in a way that protects their privacy, but without degrading application quality. For example, in a traffic speed monitoring application reconstruction of community statistics of interest (such as average traffic speed on different streets) should remain accurate, despite use of perturbed data ("lies" about actual speed of individual vehicles) as input to the reconstruction process. This is possible thanks to deconvolution techniques that recover the statistical distribution of the original signals, given the statistical distribution of perturbed data and the statistical distribution of noise. Solutions to this and related problems can be found in literature on privacy-preserving statistics [9]. Recently, special emphasis was given to perturbing time-series data [61], since sensor data typically comprise a correlated series of samples of some continuous phenomenon. Perturbing time-series data is challenging because correlations among nearby samples can be exploited to breach privacy. Recent results demonstrate that the frequency spectrum of the perturbation signal must substantially overlap with the frequency spectrum of the original data time-series for the latter to be effectively concealed [61]. Generalizations to perturbation of correlated multi-dimensional time-series data were proposed in [133]. The main challenge addressed in this work was to account for the fact that data shared by different sensors are usually not independent. For example, temperature and location data can be correlated, allowing an attacker to make inferences that breach privacy by exploiting cross-sensor correlations.

A related interesting problem is that of perturbation (i.e., noise) energy allocation. Given a perturbation signal of a particular energy budget (dictated perhaps by reconstruction accuracy requirements), how to allocate this energy budget across the frequency spectrum to optimally conceal an original data signal? A recent technique defines privacy as the amount of mutual information between the original and perturbed signals. Optimality is defined as perturbation that minimizes the upper bound on such (leaked) mutual information. The technique describes how optimal perturbation is computed, and demonstrates the fundamental trade-off between the bound on information leak (privacy) and the bound on reconstruction accuracy [132]. We note that the privacy protection issues for social sensing data arise both during trajectory data collection, and trajectory data management [38]. Since this section is focussed only on the data collection and system design issues, we will discuss this issue in a more holistic and algorithmic way in a later section of this chapter.

3.2 Generalized Model Construction

Many initial participatory sensing applications, such as those giving rise to the above privacy concerns, were concerned with computing community statistics out of individual private measurements. The approach inherently assumes richly-sampled, low-dimensional data, where many low-dimensional measurements (e.g., measurements of velocity) are redundantly obtained by individuals measuring the same variable (e.g., speed of traffic on the same street). Only then can good statistics be computed. Many systems, however, do not adhere to the above model. Instead, data are often high-dimensional, and hence sampling of the high-dimensional space is often sparse. The more interesting question becomes how to generalize from high-dimensional, sparsely-sampled data to cover the entire input data space? For instance, consider a fuel-efficient navigation example, where it is desired to compute the most fuel-efficient route between arbitrary source and destination points, for an arbitrary vehicle and driver. What are the most important generalizable predictors of fuel efficiency of current car models driven on modern streets? A large number of predictors may exist that pertain to parameters of the cars, the streets and the drivers. These inputs may be static (e.g., car weight and frontal area) or dynamic (e.g., traveled road speed and degree of congestion). In many cases, the space is only sparsely sampled, especially in conditions of sparse deployment of the participatory sensing service. It is very difficult to predict *a priori* which parameters will be more telling. More importantly, the key predictors

might differ depending on other parameters. For example, it could be that the key predictors of fuel efficiency for hybrid cars and gas-fueled cars are different. It is the responsibility of the model construction services to offer not only a general mechanism for applications to build good models quickly from the data collected, but also a mechanism for identifying the scope within which different predictors are dominant. A single "one-size-fits-all" prediction model, computed from all available data, is not going to be accurate. Similarly computing a model for each special case (e.g., a model for each type of car) is not going to be useful because, as stated above, the sampling is sparse. Hence, it is key to be able to generalize from experiences of some types of vehicles to predictions of others. Recent work combined data mining techniques based on regression cubes and sampling cubes to address the model generalization problem for sparse, high-dimensional data [64].

3.3 Real-time Decision Services

Ultimately, a generalized model, such as that described above, may be used as an input to an application-specific optimization algorithm that outputs some *decisions* for users in response to user queries. For example, estimates of fuel consumption on different roads on a map can be input to Dijkstra's algorithm to find the minimum fuel route between points specified by the user. This route constitutes a decision output. Hence, support for real-time stream processing and decision updates must be provided as part of the social sensing architecture.

A key property of real-time decision services is the involvement of humans in the loop. A significant challenge is therefore to design appropriate user interfaces. End-user devices will act as data custodians who collect, store, and share user data. The level at which these custodians interact with the user, as well as the nature of interactions, pose significant research problems with respect to minimizing inconvenience to the user while engaging the user appropriately. Context sensing, collaborative learning, persuasion, and modeling of socio-sensing systems (with humans in the loop) become important problems. Participation incentives, role assignment, and engagement of users in modeling and network learning become important application design criteria that motivate fundamental research on game theoretic, statistical, machine learning, and economic paradigms for application design.

3.4 Recruitment Issues

The quality of the social experience gained from a sensor-based framework is dependent on the ability to recruit high quality participants for

sensor collection and sharing. Many sensing systems such as those in geo-tagging applications user completely *open* frameworks in which any participant who wishes to contribute is allowed to join the sensing environment. This may of course result in numerous issues in terms of the quality of the final results:

- The participants who join may not be sufficiently trustworthy. This may impact the quality of the results. We will formally discuss the issue of trust in a later section.
- The act of participants *choosing to join* may bias the final variables which are being tracked by the application. For example, when the sensing is used in order to obtain feedback of a particular type, urban participants are more likely to join because of greater prevalence of smart phones. This kind of skew may also affect the quality of the final results.

We note that when the recruitment is performed at the initiative of the designer of the sensing system, a greater amount of control is achieved. This tends to reduce the self-selection bias, which is naturally inherent in a purely voluntary system. The work in [143] observes that the process of recruiting volunteers for participatory sensing campaigns is analogous to recruiting volunteers or employees in non-virtual environments. This similarity is used in order to create a three stage process for recruitment:

- **Qualifier:** This refers to the fact that the participants must meet minimum requirements such as availability and reputation. This ensures that high quality responses are received from the participants.
- **Assessment:** Once participants that meet minimum requirements are found, the recruitment system then determines which candidates are most appropriate based on both diversity and coverage. This ensures that bias is avoided during the recruitment process.
- **Progress Review:** Once the sensing process starts, the recruitment system must check participants' coverage and data collection reputation to determine if they are consistent with their base profile. This check can occur periodically, and if the similarity of profiles is below a threshold, this is used as a feedback to an additional recruitment process.

We note that the above process is quite similar to that of an employee hiring process in an organization, which is designed to maximize diversity, reduce bias, and maximize quality of the volunteers.

3.5 Energy Efficient Design

Since sensors often have limited battery power, it is critical to design the participatory sensing systems in order to maximize the life of the system. Many critical sensing components such as GPS tend to consume a lot of power, and can therefore result in a short life-cycle for the system. A number of tricks can be used in order to reduce the energy consumption, which may involve a reduction in the sampling rate at which the data is collected. This reduction in the sampling rate can often be achieved with the use of side-information which is collected through more efficient means. Some examples of common tricks which are used in order to improve the energy efficiency are as follows:

- In the *Sensloc* system proposed in [97], a GPS device, an accelerometer, and a WiFi scanner are simultaneously used in order to detect particular variables such as location with good accuracy. We note that the energy requirement of different kinds of sensors may vary, and the accuracy of the different sensors may also vary in a time dependent way. The typical tradeoff is that while GPS is extremely power hungry, it is also more accurate. Therefore, at a given time, the data is selectively sampled at varying rates from different sensors in order to fuse the measurements together, and provide an accurate estimation of the desired variable without too much energy consumption.

- A rate-adaptive positioning system known as *RAPS* is proposed in [129]. The system is based on the observation that GPS is generally less accurate in urban areas, and therefore it makes sense to turn it on only as often as necessary to achieve this accuracy. The *RAPS* system uses the location-time history of the user to estimate user velocity and adaptively turn on GPS only if the estimated uncertainty in position exceeds the accuracy threshold. It also estimates user movement using a duty-cycled accelerometer, and utilizes Bluetooth communication to reduce position uncertainty among neighboring devices. In addition, the system employs celltower-RSS blacklisting to detect GPS unavailability, in which case it is not turned on at all. Thus, the use of context-sensitive information in order to adaptively turn on GPS results in a considerable about of power savings [129].

- Often, a large amount of rich context sensitive information is available, which can be used to improve the accuracy of the sensing measurements without spending additional energy. For example, if the last known GPS location is overlaid on a map, then the fu-

ture location within a specific time period may be limited both by the time elapsed and the layout of that location. This can be used to estimate the location more accurately with the use of a smaller number of samples [40].

- One major issue, which has been observed in [183] is that the requests for GPS may come from Location Based Applications (LBA), and therefore, it is critical to design the energy saving strategies in the context of the LBA requests, which may not necessarily be synchronized with one another. The framework uses four design principles corresponding to *substitution*. Substitution makes use of alternative location-sensing mechanisms, such as network-based location sensing, which consume less power than GPS. Suppression uses less power-intensive sensors such as an accelerometer to suppress unnecessary GPS sensing for a statitionary user. Piggybacking synchronizes the location sensing requests from multiple running LBAs. Adaptation aggressively adjusts system-wide sensing parameters such as time and distance, when battery level is low.

- In addition to software solutions, it is also possible to implement hardware solutions. For example, simple operations can be directly performed in main memory with dedicated hardware, without actually using the (more energy-intensive) main processor [135].

In addition to the power efficiency of the *sensing* process, issues also often arise about the power-efficiency of the *data transmission* process. Typically, data transmission is significantly more expensive, as compared to the sensing process itself. For example, many applications are enabled by the ability to capture videos on a smartphone and to have these videos uploaded to an internet connected server. This capability requires the transfer of large volumes of data from the phone to the infrastructure. Typically smartphones have multiple transfer interfaces such as 3G, Edge, and Wifi, all of which vary considerably in terms of availability, data transfer rates and power consumption. In many cases, the underlying applications are naturally delay-tolerant, so that it is possible to delay data transfers until a lower-energy WiFi connection becomes available. This tradeoff is explored in some detail in the *SALSA* system proposed in [137]. An online algorithm is proposed, which can automatically adapt to channel conditions and it requires only local information to decide whether and when to defer a transmission. Such an approach has been shown to result in considerable power savings without significantly affecting the operation of the underlying system.

3.6 Other Architectural Challenges

Proper design of the above system components gives rise to other important challenges that must be solved in order to enable development and deployment of successful mobile sensing applications that adequately meet user needs. The following relates challenges described in a recent NSF-sponsored report[1] on social sensing.

From the application perspective, mobile sensing applications depend significantly on social factors (user adoption, peer pressure, social norms, social networks, etc) as well as the nature of physical phenomena being monitored or controlled. Exciting interdisciplinary research challenges exist in describing the properties of distributed socio-physical applications. For example, what are the dynamics of information propagation in such systems? What are closed-loop properties of interaction involving social and physical phenomena? What are some fundamental bounds on capacity, delivery speed, and evolution of socio-sensing systems? Answering such questions is fundamental to informed design and performance analysis of sensing applications involving crowd-sourcing.

From the underlying physical network perspective, mobile sensing applications herald an era where many network clients are embedded devices. This motivates the investigation of a network architecture, where the main goal from networking shifts from offering a mere communication medium to offering *information distillation services*. These services bridge the gap between myriads of heterogeneous data feeds and the high-level human decision needs. In a network posed as an information service (as opposed to a communication medium), challenges include division of responsibilities between the end-device (e.g., phone) and network; paradigms for data collection on mobile devices, architectural support for data management, search, and mining; scalability to large-scale real-time information input and retrieval; improved context-awareness; support for predictability; and investigation of network and end-system support for reduction of cognitive overload of the information consumer. Other challenges in the design of network protocols for mobile sensing include energy management, integration of network storage, personalized search and retrieval, support for collaborative sensing, and exploitation of a rich realm of options in information transfer modalities and timing, including deferred information sharing and delay-tolerant communication.

[1] National Science Foundation Workshop Report on Future Directions in Networked Sensing Systems: Fundamentals and Applications, The Westin Arlington Gateway, Arlington, VA, November 12-13, 2009.

While several social sensing applications are already deployed, exciting research opportunities remain in order to help understand their emergent behavior, optimize their performance, redesign the networks on which they run, and provide guarantees to the user, such as those on bounding unwanted information leakage.

4. Privacy Issues in Social Sensing

Social sensing offers interesting new challenges pertaining to privacy assurances on data. General research on privacy typically focuses on electronic communication as opposed to ramifications of increasing sensory instrumentation in a socio-physical world. In contrast, traditional embedded systems research typically considers computing systems that interact with physical and engineering artifacts and belong to the same trust domain. A need arises to bridge the gap in privacy research by formulating and solving privacy-motivated research challenges in the emerging social sensing systems, where users interact in the context of social networks with embedded sensing devices in the physical world.

Sharing sensor data creates new opportunities for loss of privacy (and new privacy attacks) that exploit physical-side channels or a priori known information about the physical environment. Research is needed on both privacy *specification* and *enforcement* to put such specification and enforcement on solid analytic foundations, much like specification and enforcement of safety requirements of high-confidence software.

Specification calls for new physical privacy specification interfaces that are easy to understand and use for the non-expert. *Enforcement* calls for two complementary types of privacy mechanisms; (i) *protection mechanisms from involuntary physical exposure*, and (ii) *control of voluntary information sharing*. The former enforce *physical privacy*. They are needed to prevent "side-channel" attacks that exploit physical and spatio-temporal properties, characteristic of embedded sensing systems, to make inferences regarding private information. Control of voluntary information sharing must facilitate privacy-preserving exchange of time-series data. A predominant use of data in social sensing applications is for aggregation purposes such as computing statistical information from many sources. Mathematically-based data perturbation and anonymization schemes are needed to hide user data but allow fusion operations on perturbed or partial data to return correct results to a high degree of approximation.

While privacy-preserving statistics and privacy-preserving data mining are mature fields with a significant amount of prior research, sharing of sensor data offers the additional challenge of dealing with *cor-*

related multi-dimensional time-series data represented by sensory data streams. Correlations within and across sensor data streams and the spatio-temporal context of data offer new opportunities for privacy attacks. The challenge is to perturb a user's sequence of data values such that (i) the individual data items and their trend (i.e., their changes with time) cannot be estimated without large error, whereas (ii) the distribution of the data aggregation results at any point in time is estimated with high accuracy. For instance, in a health-and-fitness social sensing application, it may be desired to find the average weight loss trend of those on a particular diet or exercise routine as well as the distribution of weight loss as a function of time on the diet. This is to be accomplished without being able to reconstruct any individual's weight and weight trend without significant error.

Examples of data perturbation techniques can be found in [14, 13, 59]. The general idea is to add random noise with a known distribution to the user's data, after which a reconstruction algorithm is used to estimate the distribution of the original data. Early approaches relied on adding independent random noise. These approaches were shown to be inadequate. For example, a special technique based on random matrix theory has been proposed in [95] to recover the user data with high accuracy. Later approaches considered hiding individual data values collected from different private parties, taking into account that data from different individuals may be correlated [86]. However, they do not make assumptions on the model describing the *evolution* of data values from a given party over time, which can be used to jeopardize privacy of data streams. Perturbation techniques must specifically consider the data evolution model to prevent attacks that extract regularities in correlated data such as spectral filtering [95] and Principal Component Analysis (PCA) [86]. In addition to data perturbation, numerous *group-based anonymization* methods have been proposed such as k-anonymity and ℓ-diversity [9]. In k-anonymity methods, the data features are perturbed, so that adversarial attacks always retain an ambiguity level over k-different participants. In ℓ-diversity, criteria are imposed over a group to ensure that the values of the sensitive attributes are sufficiently diverse within a group. This is motivated by the observation that k-anonymity may sometimes not preserve the truth about individual sensitive values, when all sensitive values within an anonymized group are the same.

In work discussed earlier in this chapter [61], it was shown that privacy of time-series data can be preserved if the noise used to perturb the data is itself generated from a process that approximately models the measured phenomenon. For instance, in the weight watchers example, we may have an intuitive feel for the time scales and ranges of weight

evolution when humans gain or lose weight. Hence, a noise model can be constructed that exports realistic-looking parameters for both the direction and time-constant of weight changes. The resulting perturbed stream can be aggregated with that of others in the community. Since the distributions of noise model parameters are statistically known, it is possible to estimate the sum, average and distribution of added noise (of the entire community) as a function of time. Subtracting that known average noise time series from the sum of perturbed community curves will thus yield the true community trend. The distribution of community data at a given time can similarly be estimated (using de-convolution methods) since the distribution of noise (i.e., data from virtual users) is known. The estimate improves with community size.

The approach preserves individual user privacy while allowing accurate reconstruction of community statistics. Several research questions arise that require additional work. For example, what is a good upper bound on the reconstruction error of the data aggregation result as a function of the noise statistics introduced to perturb the individual inputs? What are noise generation techniques that minimize the former error (to achieve accurate aggregation results) while maximizing the noise (for privacy)? How to ensure that data of individual data streams cannot be inferred from the perturbed signal? What are some bounds on minimum error in reconstruction of individual data streams? What noise generation techniques maximize such error for privacy? Privacy challenges further include the investigation of attack models involving corrupt noise models (e.g., ones that attempt to deceive non-expert users into using perturbation techniques that do not achieve adequate privacy protection), malicious clients (e.g., ones that do not follow the correct perturbation schemes or send bogus data), and repeated server queries (e.g., to infer additional information about evolution of client data from incremental differences in query responses). For example, given that it is fundamentally impossible to tell if a user is sharing a properly perturbed version of their real weight or just some random value, what fractions of malicious users can be accommodated without significantly affecting reconstruction accuracy of community statistics? Can damage imposed by a single user be bounded using outlier detection techniques that exclude obviously malicious users? How does the accuracy of outlier detection depend on the scale of allowable perturbation? In general, how to quantify the tradeoff between privacy and robustness to malicious user data? How tolerant is the perturbation scheme to collusion among users that aims to bias community statistics? Importantly, how does the *time-series* nature of data affect answers to the above questions com-

pared to previous solutions to similar problems in other contexts (e.g., in relational databases)?

Furthermore, how can the above perturbation techniques, defense solutions, and bounds be extended to the sharing of multiple correlated data streams, or data streams with related context? For example, consider a social sensing application where users share vehicular GPS data to compute traffic speed statistics in a city. In this case, in order to compute the statistics correctly as a function of time and location, each vehicle's speed must be shared together with its current GPS location and time of day. Perturbing the speed alone does not help privacy if the correct location of the user must be revealed at all times. What is needed is a perturbation and reconstruction technique that allows a user to "lie" about their speed, location, and time of day, altogether, in a manner that makes it impossible to reconstruct their true values, yet allow an aggregation service to average out the added multi-dimensional noise and accurately map the true aggregate traffic speed as a function of actual time and space. This problem is related to the more general concern of privacy-preserving classification [158, 176], except that it is applied to the challenging case of aggregates of time-series data. Other methods for centralized and distributed privacy preservation in time series include the methods discussed in [130, 141], though these methods are generally *offline*, and cannot easily perform the privacy preservation in real time, as would be needed for a typical social sensing application.

In many participatory sensing applications, users may upload different kinds of data such as images, text, or other feeds to the system. Such data are often tagged with location (WiFi or GPS) and the time-stamp, which can have serious consequences in terms of location privacy. Alternatively, the users may have to continuously provide their location to an untrusted service provider, or provide responses to queries which may compromise their privacy. Some of the earliest work on location privacy [152] focusses only on user identity suppression, while preserving the full fidelity of the location data. This approach of course suffers from the well known problem of adversarial attacks with background information about approximate location. The work in [66, 75, 94, 131] avoids this pitfall by using a k-anonymity approach for the spatio-temporal scenario. The work in [94] proposed a technique called *tessellation*, in which a point location is enlarged to a tile which contains at least k users. This is essentially a spatio-temporal version of the *generalization* technique which is often used in k-anonymity applications. It was observed in [87], that tessellation is not useful in applications where the large tiles do not provide the fine grained information about the location for a particular user (such as the road information). Therefore, the work in [87]

uses a clustering (micro-aggregation) approach, which is able to preserve more fine grained information about the location. In this context, the method of [2] also treats the trajectory of an object as a cylinder in 3-dimensional space, where the radius of the cylinder is non-zero because of the uncertainty in the GPS position of the object. The key here is to understand is that the uncertainty is inherent to the method of collecting the data, since all GPS collection methods have a certain level of error associated with them. In this context, the work in [2] defines the concept of (k, δ)-anonymity, which is a set S of at least k trajectories, such that all of these trajectories lie within a distance of at most $\delta/2$ of the average position of these different trajectories. We note that it may not be possible to create $(k.\delta)$-anonymized groups from the original data set, if some of the trajectories are somewhat isolated. Therefore, the work in [2] proposes the *Never Walk Alone (NWA)* algorithm, in which the positions of some of the objects is distorted with *space translation*, so that it is possible to construct such (k, δ)-anonymized groups from the data. The approach constructs these anonymized groups while minimizing the total distortion in the data.

Many mobile applications can infer the *context* of a user from GPS (e.g. whether a user is at home or work). It has become increasingly common for many mobile applications to aggressively collect such context data [56] for a variety of applications. Such context can sometimes be very sensitive from a release perspective. For example, a user may not wish anyone to know whether they are currently in a hospital. The afore-mentioned k-anonymization does not necessarily help protect the sensitivity of context, if all of the k users within a group are at the same sensitive location. A number of methods use full suppression techniques [83, 157] in which the location or context of the user is suppressed when they are at a sensitive location. However, it has been observed in [74] that the *fact of the suppression itself* can be sensitive information, in the presence of a powerful adversary with greater background knowledge.

Another issue with mobile sensing applications is that considerable temporal correlations exist between the different locations of a single or multiple users. Such correlations can be used in order to perform privacy attacks which can infer the sensitive locations of different users. In this context, a number of methods [30, 68, 69, 76, 126] have been designed which utilize the temporal correlations in the privacy preservation process. The work in [76] observes that one can use linear interpolation to infer suppressed locations. Therefore, the work in [76] works by constructing zones which contain multiple sensitive locations, and the anonymization process introduces a sufficient amount of uncertainty in each zone. It has been observed in [30] that information about the veloc-

ity of a user can be used in order to infer their location during successive time instants. For example, for two successive zones containing a user, the velocity of the user provides implicit limits on where they may or may not be found at any given time. The work in [30] protects against such kinds of privacy attacks. The work in [68] improves these methods by introducing temporal delays. However, none of these methods can provably protect privacy, when an adversary knows the system that is used for anonymization. The work in [74] designs a scheme which can preserve the privacy of sensitive user locations in the presence of such powerful background knowledge.

Location privacy systems can also be understood in terms of *Quality of Service (QoS)* models in response to user location queries. Such models consider the fact that the use of generalization (eg. spatial and temporal cloaking) and suppression (eg. dropping a trajectory from query output) for privacy preservation reduces the accuracy of responses to user-queries. Therefore, a significant amount of research has also been focussed on performing the privacy-preservation with a focus on maintaining certain levels of QoS for privacy preservation [17, 67, 125, 149]. These methods generally work with optimizing common models for k-anonymity and ℓ-diversity, with a specific focus on improving the QoS for user queries.

Finally, it has been recognized, that in many mobile sensing applications, it is not required to collect the individual sensor streams, but one may only desire to compute the aggregate statistics from these sensors. For example, many location-based vehicular services are designed into the national transportation infrastructure in many countries. These include usage- or congestion-based road pricing, traffic law enforcement, traffic monitoring, and vehicle safety systems. Such applications often require the computation of aggregate statistics, but poorly chosen implementations can result in violations of privacy. For example, the GPS monitoring of cars as they arrive, or the use of surveillance cameras and toll transponders can result in privacy violations.

In the context of such applications, the following functionalities need to be provided:

- In many applications, some centralized server needs to compute a function of a car's *path*, which is essentially a list of time-position tuples. A system called *VPriv* [134] provides a protocol to compute path functions in a way, such that it does not reveal anything more than the result of the function to the server. In addition, an enforcement mechanism is provided (using random spot checks) that allows the server and application to handle misbehaving cars.

- The *PrivStats* system computes aggregate statistics on user location information and guarantees location privacy even in the face of side information about user location and movement patterns. It is also resistant to large amounts of spurious data upload by users.

Many applications require the computation of a specific function on the data, and therefore, it is critical to design methods for computing the function accurately on the perturbed data. For example, the problem of privacy-preserving regression modeling of sensor data has been discussed in [3].

5. Trust in Social Sensing

At the broadest level, social sensing systems can be considered multi-agent systems, that interact with one another and provide a variety of data-centric services to one another. Therefore, a number of issues of trust arise in the context of such large-scale social-centric applications, which are common to many traditional peer-to-peer applications [138]. Such issues typically deal with the the aspect of designing trustworthy protocols for interactions between different agents, both in terms of the choice of interactions, and the time of these interactions. A detailed survey of the (more traditional) literature along this direction may be found in [138]. The more recent social sensing work has focussed on the *data-centric* aspects of trust, rather than the *interaction-centric* aspects.

The openness of participatory sensing systems provides them with a tremendous amount of power in collecting information from a wide variety of sources, and distilling this information for data mining purposes. However, it is this very openness in data collection, which also leads to numerous questions about the quality, credibility, integrity, and trustworthiness of the collected information [45, 51, 71, 72]. Furthermore, the goals of privacy and trust would seem to be at odds with one another, because all privacy-preservation mechanisms *reduce* the fidelity of the data for the end-user, whereas the end-user trust is dependent on *high* fidelity of the data. Numerous questions may arise in this respect:

- How do we know that the information available to the end user is correct, truthful and trustworthy?
- When multiple sources provide conflicting information, how do we know who to believe?
- Have errors been generated in the process of data collection, because of inaccuracy or hardware errors?

The errors which arise during hardware collection are inherent to the device used, and their effect can be ameliorated to some extent by care-

ful design of the underlying application. For example, the *LiveCompare* [46] application (described in detail in the application section), which is used for comparison shopping of grocery products, works by allowing individuals to transmit photographs taken in stores of grocery products, and then presents similar pictures of products taken in nearby stores. The approach allows the transmitting of product photos taken by individual users of competing products, but does not automatically try to extract the pricing information from the price tags in the photograph. This is because the extraction process is known to be error-prone, and this design helps avoid the inaccuracy of reporting the pricing of competing products. It also avoids manual user input about the product which reduces error and maximizes trustworthiness.

For the case of specific kinds of data such as *location data*, a variety of methods can be used in order to verify the truthfulness of the location of a mobile device [107]. The key idea is that time-stamped location certificates signed by wireless infrastructure are issued to co-located mobile devices. A user can collect certificates and later provide them to a remote party as verifiable proof of his or her location at a specific time. The major drawback of this approach is that the applicability of these infrastructure based approaches for mobile sensing is limited as cooperating infrastructure may not be present in remote or hostile environments of particular interest to some applications. Furthermore, such an approach can be used only for particular kinds of data such as location data.

In the context of participatory sensing, where raw sensor data is collected and transmitted, a basic approach for ensuring the integrity of the content has been proposed in [51], which guards whether the data produced by a sensor has been maliciously altered by the users. Thus, this approach relies on the approach of *platform attestation* which vouches that the software running on the peripheral has not been modified in an unintended manner. This kind of approach is more useful for sensors in which the end data is produced by the device itself, and an automated software can be used for detection of malicious modification. In essence, the approach allows the trusted sensing peripherals to sign their raw readings, which allows the remote entity to verify that the data was indeed produced by the device itself and not modified by the user.

An additional challenge which naturally arises in the context of data trustworthiness is that the goals of data integrity and authenticity run contrary to the goals of user privacy. Almost all privacy-preserving data mining algorithms reduce the data fidelity in some way in order to reduce the ability to identify sensitive information about the user. Clearly,

such an approach will not work in the context of systems such as those proposed in [51].

Trusted Platform Module (TPM) hardware [71], commonly provided in commodity PCS, can be leveraged to help provide this assurance. To address the problem of protecting the privacy of data contributors, techniques such as requiring explicit authorization for applications to access local resources and formulating and enforcing access control policies can be used. A TPM is a relatively inexpensive hardware component used to facilitate building trusted software systems. It is possible to leverage the TPM functionality of attesting to the integrity of software running on a device to a remote verifier. The TPM can attest to the software platform running on the machine by providing a signed quote of its PCR(s) in response to a challenge from a remote verifier.

In many cases, user actions may change the data (such as the cropping of an image), but this may not actually affect the trust of the underlying data. The work in [72] proposes *YouProve*, which is a partnership between a mobile device's trusted hardware and software that allows un-trusted client applications to directly control the fidelity of data they upload and services to verify that the meaning of source data is preserved. The approach relies on trusted analysis of derived data, which generates statements comparing the content of a derived data item to its source. For example, the work in [72] tests the effectiveness of the method on a variety of modifications on audio and photo data, and shows that it is possible to verify which modifications may change the meaning of the underlying content.

A more critical question about trustworthiness arises when the data is collected through the actions of end users. In such cases, the user responses may have an inherent level of errors which may need to be evaluated for their trustworthiness. The issue of truthfulness and trust arises more generally in any kind of application, *where the ability to contribute information is open*. Such openness is a double-edged sword, in that it greatly increases information availability at the expense of trust. Aside from social and participatory sensing platforms, any *web-enabled platforms* which allow the free contribution of information may face such issues.

In this context, the problem of trustworthiness has been studied for resolving multiple, conflicting information provision on the web. The earliest work in this regard was proposed in [170], where the problem of studying conflicting information from different providers was studied [170]. Subsequently, the problem of studying trustworthiness in more general dynamic contexts was studied in [48, 49].

A number of recent methods [103, 159–162] address this issue, in which a consistency model is constructed in order to measure the trust in user responses in a participatory sensing environment. The key idea is that untrustworthy responses from users are more likely to be different from one another, whereas truthful methods are more likely to be consistent with one another. This broad principle is used in order to model the likelihood of participant reliability in social sensing with the use of a Bayesian approach [159, 160, 162]. A system called *Apollo* [103] has been proposed in this context in order to distill the likely truth from noisy social streams.

Such social streams are also often used in the context of applications, where *alarms* are raised in response to specific events. The nature of the alarm may vary with the application scenario. For example, in a military network, the alarm may be raised because of enemy threats, whereas in a patient monitoring application, the alarm may be raised because of a medical emergency. Such applications are inherently error prone and raise many false alarms because of technology limitations. For example, errors in the collection of the sensor readings, or an innocuous activity may trigger a false alarm. In [153], the problem of trustworthiness of such alarms has been studied, and a number of methods have been proposed in order to provide more accurate and trustworthy alarms.

6. Implied Social Networks: Inference and Dynamic Modeling

In the case of an explicitly linked social network, the relationships between different entities are quite clear, and therefore the dynamics of the interaction can be modeled relatively easily. However, in the case of a participatory sensing environment, the links between different entities may change *rapidly and dynamically*. Furthermore, such links may either be *explicit* or *implicitly derived* based on the dynamic interactions between participants. For example, the *Google Latitude* application allows for explicit links between different agents. On the other hand, in many social applications [52, 34], the links and communities between different agents may need to be derived based on their location and behavior. In such cases, the structure of the social network itself and the underlying communities [53, 35, 36, 163] can be derived directly from the details of the underlying interaction. This is a challenging problem, especially when the number of agents are large, and the number of interactions between them is even larger and dynamically evolving. Furthermore, a variety of context-specific information such as organizational rhythms, socially significant location and daily activity patterns may need to be

simultaneously derived and used [52] for inferring the significant links. The work in [44] derives the links between users based on their mobility patterns from GPS trajectories. In order to achieve this goal, the work in [44] divides the spatial regions into a grid, and constructs nodes for each cell. An edge exists between a pair of nodes, if a trajectory exists which starts at one cell and ends at another. By performing the discretization at varying levels of granularity, it is possible to analyze different characteristics of the underlying users. The work in [44] specifically shows how the approach can be used for effective community detection.

An interesting work in [173] examines the common patterns in the activities of different geo-tracked users, and makes friendship or linkage recommendations on the basis of significant overlaps in activity patterns. It has also been observed in [115] that different kinds of sharing in activity patterns may have different significance for different users. For example, it is possible that two individuals that are friends may not spend a lot of time together, but only a couple of hours on a Saturday night. On the other hand, a pair of co-workers who are not friends may share a lot of time together. Thus, it is critical to be able to learn the importance of different kinds of commonality in patterns in the prediction process [115]. Such trajectory analysis is useful not just for determining useful relationships, but also interesting places, travel sequences or activities which are relevant to such relationships [27, 181]. In particular, an interesting *authority-based* model for relating social behavior and location behavior has been proposed in [27]. The essential idea is to construct a graph which models relationships of the trajectories of the different users to the different locations. The idea is that authoritative users are also likely to visit authoritative places and vice-versa. This is used in order to construct a page-rank like model in order to determine both the authoritative users and authoritative locations simultaneously.

Many sensing platforms such as those discussed in [33], yield sensor data which is varied, and is of a multi-modal nature. For example, the data could contain information about interactions, speech or location. It is useful to be able analyze such data in order to summarize the interactions and make inferences about the underlying interactions. Such multi-modal data can also be leveraged in order to make predictions about the nature of the underlying activities and the corresponding social interactions. This also provides a virtual technique to perform link inferences in the underlying network.

The collection of activity sensing data is not very useful, unless it can be leveraged to summarize the nature of the activities among the different participants. For example,in the case of the techniques discussed in [34], the IR transceiver is used to determine which people are in prox-

imity of one another. However, this cannot necessarily be used in order to determine whether the corresponding people are interacting with another. A knowledge of such interactions can be determined with the use of *speech segmentation* techniques in which it is determined which participants are interacting with one another. The speech portions are segmented out of the ambient noise, and then segmented into conversations. The knowledge of such face-to-face interactions can be used to build dynamic and virtual links among the different participants.

We note that a dynamically linked social network can be modeled in two different ways:

- The network can be modeled as a group of dynamic interacting agents. The stochastic properties of these agents can be captured with the use of hidden markov models in order to characterize various kinds of behaviors. This is the approach used for community modeling as discussed in [15, 36].
- The interactions of the participants can be modeled as links which are continuously created or destroyed depending upon the nature of the underlying interactions. as a graph stream, in which the nodes represent the participants, and the edges represent the interactions among these different participants. Recently, a number of analytical techniques have been designed in order to determine useful knowledge-based patterns in graph streams [8]. These include methods for dynamically determining shortest-paths, connectivity, communities or other topological characteristics of the underlying network.

The inherently dynamic nature of such interactions in an evolving and dynamic social network leads to a number of interesting challenges from the perspective of social network analysis. Some examples of such challenges are discussed below.

(1) Determination of dynamic communities in graph streams: Communities are defined as dense regions of the social network in which the participants frequently interact with one another over time. Such communities in a dynamically evolving social network can be determined by using agent-based stochastic analysis or link-based graph stream analysis. Methods for modeling such a social network as a group of dynamically evolving agents are discussed in [15, 36]. In these techniques, a hidden markov model is used in conjunction with an influence matrix in order to model the evolving social network.

A second approach is to model the underlying face-to-face interactions as dynamic links. This creates an inherently dynamic network scenario in which the structure of the communities may continuously evolve over

time. Therefore, a key challenge is to determine such communities in dynamic networks, when the clustering patterns may change significantly over time. Methods for determining evolving clusters and communities in networks have been discussed in [10, 12, 31, 28, 79, 151]. Many of these methods determine communities in the underlying data by incorporating concepts of *temporal smoothness*, wherein the structure of the communities is allowed to evolve only in a smooth way over time. On the other hand, when the data is of very high volume (such as a graph stream), it is also critical to design very efficient methods for community maintenance. Graph streams pose a special challenge because of the rapid nature of the incoming edges, and their use for determination of evolving communities.

(2) Mining Structural Patterns in Time-Evolving Social Networks: Aside from the common problem of community detection, another interesting problem is that of mining structural patterns of different kinds in time evolving graphs. Some common methods for finding such patterns typically use matrix and tensor-based tools, which are comprehensively described in a tutorial in [60]. Common problems in time-evolving graphs include those of frequent pattern determination, outlier detection, proximity tracking [156], and subgraph change detection [118].

(3) Modeling spatio-temporal dynamics: Many of the approaches discussed above model the dynamics of the interactions as dynamic links. While this provides greater generality, it does not capture the spatio-temporal nature of the underlying agents. For example, the data received in a GPS application often contains spatio-temporal information such as the positions of different agents, and their underlying interactions. Therefore, an interesting and important challenge is to model the aggregate spatio-temporal dynamics in order to determine the underlying patterns and clusters. Such spatio-temporal dynamics can be used in order to make interesting *spatial predictions* such as future regions of activity or congestions. Many methods for clustering, community detection, classification, and outlier detection from such data have been proposed in [104, 105, 112–115, 109, 110] and are discussed in some detail in the application section of this chapter. In many cases, such data may even be combined with other content-based data such as GPS-tagged images and documents in order to further improve the quality of the underlying inference [172].

(4) Modeling Influential Community Members: This problem is essentially that of determining the members of the participatory sensor network, who have the greatest influence on their peers in the community. Alternatively, it may also be interesting to trace back the spread of

rumors or other information in the community. In a static network such as *Facebook*, the problem of influence analysis is much more straightforward, because it depends upon the static connections between the different communities [96]. In a dynamic network, the underlying network structure may change rapidly over time, depending upon the interactions between the underlying entities. Some recent work on dynamic influence analysis addresses this scenario of interactions between dynamic and evolving entities [11]. This method can determine either influential nodes or determine the most likely points of release, based on a given influence pattern and also a given pattern of interactions. A classic example of a dynamic network in the context of social sensing is the *face-to-face interaction network*, in which it may be desirable to determine the influence of such interactions on specific behaviors. For example, the work in [122] used a mobile phone-based sensing platform to examine the influence of face-to-face interactions in the life-style choices of participants such as obesity, eating and exercise habits. It was shown that the use of sensing platforms can be very effective at modeling the influence effects of such interactions (which turned out to be significant for this scenario).

As discussed earlier, the determination of dynamic interactions can sometimes require the real-time modeling of *implied interactions* (such as face-to-face interactions), which are hard to infer from sensor data can also sometimes be sensitive information. This also leads to numerous privacy challenges, especially since the interactions between the participants may be considered personal information. As mentioned earlier, privacy continues to be an important issue for such social sensing applications. A number of privacy-sensitive approaches for face-to-face activity modeling and conversation segmentation have been discussed in [164–167].

The dynamic modeling of social sensing applications, naturally lead to a lot of trajectory data in real applications. Therefore, significant amount of research has been devoted to determining spatio-temporal patterns from such trajectories. Such patterns may be derived with or without additional content information. A number of these methods will be discussed in the next section.

7. Trajectory Mining for Social Sensing

Social sensing applications have naturally lead to the collection of trajectory database from the rich GPS data, which is collected in a wide variety of applications. The increasing popularity and availability of mobile phones also enables the collection of trajectory data from willing

participants with the use of widely downloadable mobile applications, as long as appropriate privacy-protection mechanisms are in place. A classic example of such a data set is the well known *GeoLife* data set [194]. Such data sets are not just collected for humans, but even from animals for tracking purposes. An example of such an animal tracking database is the *Movebank* database [186], which contains detailed data about animal trajectories in the data. Finally, many recent document and image creation hardware such as GPS-enabled cameras and cellphones automatically stamp the content with GPS locations. This creates a very rich data set containing *both* content and (implicit) trajectories. The availability of such data makes it important to design more effective and efficient methods for trajectory mining.

Trajectory data is particularly useful from the perspective of mining aggregate community movement patterns. A variety of interesting patterns can be mined in such trajectory data sets, which provide insights into the aggregate movements. The aggregate movements are best represented by clusters, which are variously referred to as *flocks*, *convoys*, or *swarms* [22, 77, 91, 102, 104, 113], depending upon the model which is used to characterize these clusters. Typically, the goal is to either determine objects with trajectories of similar shape, or objects which move together in clusters. The major difference between these different kinds of moving clusters are as follows:

- *Flocks:* These correspond to groups of objects which move within a fixed disc of a particular size over consecutive time-stamps [77, 102]. As a result, the underlying trajectories will often have a similar geometric shape.

- *Moving Cluster:* This refers to a group of objects which have considerable overlap between successive time-stamps [93]. As in the previous cases, the constraint on the objects moving together in successive time stamps leads to trajectories of similar shape.

- *Convoys:* In this case, we again find groups of objects which move together, except that the concept of density is used in order to define the objects that move together. As before, the objects need to move together over consecutive time stamps [90, 91]. In many scenarios, the use of density provides a flexible way of modeling the movement of significant masses of objects together.

- *Swarms:* In the case of swarms, the objects are required to move together as before, except that we do not impose the requirement that the objects should be together over consecutive time stamps

[113]. In such a case, the shapes of the trajectories of the different objects may sometimes be quite different. The approach discussed in [113] first uses an off-the-shelf spatial clustering algorithm to partition the objects into clusters at each time-stamp. This transforms the spatial trajectories into data which represents membership of objects in clusters. Subsequently, a frequent pattern mining-like approach is applied to this transformed data into order to determine those objects which belong to the same spatial cluster for a significant number of time-stamps. An Apriori-like approach is used for this purpose, in combination with a number of additional pruning tricks, which use the temporal characteristics of the data. Since the consecutiveness of the membership information is not used in the pattern-mining phase, the swarms are based on significant levels of co-location at any period in time.

The problem of clustering is particularly useful from the perspective of trajectory mining, because it provides summary information which can be used for other applications. For example, the *TraClass* method proposed in [105] uses two kinds of clustering in order to provide additional summary information, which enables more effective classification. One kind uses the characteristics of different regions in the clustering, but it does not use the movement patterns. The other kind uses the characteristics of different trajectories in the clustering. The two kinds of clusters provide useful complementary information in the classification process. It has been shown in [105], how this additional information can be leveraged for a more effective classification process.

While clustering determines the *typical* movement patterns, a related problem is that of determining unusual (or *atypical*) movement patterns [105, 109, 110]. Such movement patterns are also referred to as outliers. Another variation on the problem is the determination of periodic patterns [114], which we wish to determine common patterns of movement which repeat periodically in the trajectory data, or hot routes in road networks [111]. A comprehensive range of trajectory mining techniques have been developed in the context of the *MoveMine* project at UIUC [115].

While much of this work has been performed in the context of animals, similar techniques can be generalized to the case of humans. Human movements are of course somewhat more complex, because of the greater complexity of social interactions as compared to animals. Some recent work has been performed on studying the trajectory patterns of humans, which were collected from mobile phones [73]. It was shown that human trajectories show a high degree of temporal and spatial regularity, and each individual shows a highly time-independent charac-

teristic travel distance and a significant probability to return to a few highly frequented locations. On further simplification, it was shown that individual travel patterns collapse into a single spatial probability distribution. This suggests that humans follow simple reproducible patterns. This simple observation has consequences for all phenomena driven by human mobility, such as epidemic prevention, emergency response, urban planning and agent-based modeling.

A key area of research for mobile trajectory analysis is to determine frequent and repetitive trajectories in the data. The most basic analysis from this perspective is to determine similar trajectories to a given target trajectory. A variety of methods on the topic of indexing moving object databases may be found in [80]. The problem has also been studied in the context of the gps trajectories created by mobile phones [43, 174]. A method for performing user-oriented trajectory search for trip recommendations has been proposed in [147].

More generally, the work in [70] explores the sequential pattern mining problem in the context of trajectory pattern mining. The idea is to determine sequences of places in the data, which occur together frequently in the data, and with similar transition times. The sequential pattern mining paradigm can be extended to this case by incorporating temporal constraints into successive elements of the sequence.

Trajectory patterns can also be derived from geo-tagged photos, in which users utilize gps-enabled mobile phones to take photos and upload them. Since the user location and time is recorded, when they take the photo, this provides natural way to derive the trajectory of the user. For example, the work in [171] mines frequent sequential trajectory patterns from such geo-tagged social media. However, the number of patterns may be too large to be informative to a user. Therefore, a ranking mechanism is introduced in order to determine the importance of the different reported patterns. The relationships between users, locations and patterns and their importance are utilized for ranking purposes. For example, trajectories are considered important, if they are followed by important users, and contain important locations. The vice-versa relationships also hold in this case. These importance relationships are modeled in [171] with the use of matrices representing the pairwise relationships between users, locations and patterns. A system of equations is set up with these matrices and solved in order to determine the importance values of the different trajectories. In addition, a *diversification criterion* is introduced in order to ensure that trajectories with large segments in common are not reported simultaneously. This is done in order to maximize the amount of useful in information in a small number of presented results. The GPS data can also be used in order to determine

interesting locations, trajectories, or even the transportation modes of the different users [180, 181].

While social sensing applications are generally defined for the case of people, a similar analysis can be applied to the case of online tracking of animals. For example, animals which are drawn from the same community or family may be considered to have implicit links among them. Such links can be utilized for the perspective of detailed understanding of how community and family membership affects geographical patterns. Such information can be very useful for a variety of applications, such as building disease propagation models among animals.

7.1 Integrating Sensor Data with Heterogeneous Media for Enhanced Mining and Inference

Many of the devices (such as mobile phones), which enable social sensing applications are *convergent devices*, which provide multiple functionalities in recording different kinds of media data. For example, most mobile phones today provide the capability to record photos, videos, text blogging and tweets, and upload them directly in real time. Thus, such media data automatically becomes *geo-tagged*, and this additional information provides a rich source of information for improving the mining process.

For example, the problem of providing location and activity recommendations on the basis of user contributed comments and their GPS trajectories has been studied in [179]. The user comments provide deeper insights into their activity histories, which can be leveraged for a better mining process. The collective wisdom of the trajectories and comments of different users can be leveraged in order to provide answers to questions such as the following:

- For a particular activity, what are the most appropriate places to visit?
- For a particular location, which a user has already visited, what are the other activities that can be performed at that location?

In order to achieve this goal, the user location and activity histories are used as the input. We note that the activity histories can only be indirectly derived from user comments, by mining the relevant words in the comments, which are related to specific activities. The location features and activity-activity correlations are mined in order to obtain additional knowledge. A collective matrix factorization methods was applied in [179] in order to mine interesting locations and activities and recommend them to users. Location information is also useful for rec-

ommending specific non-spatial products or items on the basis of spatial history, as discussed in [101].

This general principle can also be applied for geographical topic discovery and comparison from GPS-associated documents [172]. While topic modeling of documents is widely known, the use of geographic information in the process provides rich opportunities for adding additional insights into the process. Many interesting concepts, including cultures, scenes, and product sales, correspond to specialized geographical distributions. The goal of geographical topic discovery is to discover such interesting concepts. The two main questions in this context are as follows:

- What are the coherent topics of interest in the different geographical regions?
- How can the different topics be compared across the different geographical regions?

The work in [172] proposes and compares three different models which use pure location, pure text, and a joint model of location and text, which is referred to as LGTA (Latent Geographical Topic Analysis). The approach is used on several data sets from the *Flickr* web site. It is shown that the first two methods work in some data sets but fail in others, whereas LGTA works well in all data sets at finding regions of interest and also providing effective comparisons of the topics across different locations. This suggests that geographical data and content data provide complimentary information to one another for the mining process. Further work along this direction in the context of *topic evolution* is proposed in [169]. From a real-time perspective, it is often useful to utilize location information for providing context-sensitive newsfeeds to users [19].

An interesting application in [32] shows that the latent information in user trajectories, which are extracted from the GPS data in photos can even be used to generate travel itineraries. For example, the media sharing site *Flickr*, allows photos to be stamped by the time of when they were taken and be mapped to points of interest with the use of geographical and tag meta-data. This information can be used to construct itineraries with a two-step approach. First, the photo streams of individual users are extracted. Each photo stream provides estimates on where the user was, how long he stayed at each place, and what was the transit time between places. In the second step, all user photo streams are aggregated into a Point of Interest (POI) graph. Itineraries are then automatically constructed from the graph based on the popularity of the POIs, and subject to the user time and destination constraints.

8. Social Sensing Applications

In this section, we will discuss a number of recent applications which have been designed in the context of sensors and social networks. Many of these applications are related to storage and processing of mobile data which is continuously collected over time. Such mobile data can be used in order to provide real time knowledge of the different users to one another, trigger alerts, provide an understanding of social trends, and enable a variety of other applications. In this section, we will discuss a number of social-centric applications, which have been developed in recent years. These include specific systems which have been designed by companies such as Google, Microsoft, and SenseNetworks, as well as a number of generic applications, which have not yet been fully commercialized.

8.1 CrowdSourcing Applications for User-Centered Activities

A number of crowdsourcing applications have recently been designed for providing feedback in a number of user-centered activities such as buying behavior, location trends, or other miscellaneous user activity. Examples of such applications include *Google Latitude*, *CitySense*, *Macrosense* and *Wikitude* applications. The Citysense and Macrosense applications both collect real-time data from a variety of GPS-enabled cell phones, cell phone tower triangulation, and GPS-enabled cabs. The two applications share a number of similarities in terms of the underlying methodology, but they have different features which are targeted towards different kinds of audiences. We describe them below:

8.1.1 The Google Latitude Application. The Google Latitude Application uses GPS data which is collected from Google map users on mobile cell phones. It is also possible to collect more approximate data with the use of cell phone tower location data (in case the mobile phones are not GPS enabled), or with the use of IP addresses of a computer which is logged into the personalized google page called *iGoogle*. The Latitude application enables the creation of *virtual friends*, who are essentially other users that carry the same location-enabled device, or use other devices such as personal computers which can transmit approximate location data such as IP-addresses. A number of other applications which enabled by the Google Latitude master application are as follows:

- **Location Alerts:** The application allows the triggering of alerts when someone is near their latitude friends. The alerts are trig-

gered only when something interesting is being done. This is done on the basis of both time and location. For example, an alert could be triggered when two friends are at a routine place, but an unusual time. Alternatively, it could be triggered when two friends are at a routine time but unusual place.

- **Public Location Badge:** It is possible to post one's location directly on blog or social network. This in turn increases the visibility of one's information to other users of the site.
- **Use with Chat Applications:** The mobile location can also be used in conjunction with the *Google Talk* application which allows users to chat with one another. Users who are chatting with one another can see each other's location with the use the embedded latitude functionality.

It is clear that all of the above techniques change the nature and dynamics of social interactions between users. For example, the triggering of alerts can itself lead to a changed pattern of interaction among the different users. The ability to mine the dynamics of such interactions is a useful and challenging task for a variety of applications.

While *Google Latitude* is perhaps the most well known application, it is by no means the only one. A number of recent applications have been designed which can track mobile devices on the internet through GPS tracking. Some of these applications have been designed purely for the purpose of tracking a device which might be lost, whereas others involve more complex social interactions. Any software and hardware combination which enables this has the potential to be used for social sensing applications. Some examples of such applications are as follows:

- **Navizon Application:** This application [188] uses GPS in order to allow social interactions between people with mobile phones. It allows the tracking of mobile friends, coverage of particular areas, and trails followed by a particular user.
- **iLocalis Application:** This application [189] is currently designed only for particular mobile platforms such as the iPhone, and it allows the tracking of family and friends. In addition, it is also designed for corporate applications in which a group of mobile employees may be tracked using the web. Once friendship links have been set up, the application is capable of sending a message to the friends of a particular user, when they are nearby.

8.1.2 CitySense Application. The citysense application is designed for the broad consumer base which carries mobile cell phones.

The Citysense application is designed to track important trends in the behavior of people in the city. For example, the application has been deployed in San Francisco, and it can show the busiest spots in the city on a mobile map.

The CitySense application also has a social networking version of a collaborative filtering application. The application stores the personal history of each user, and it can use this personal history in order to determine where other similar users might be. Thus, this can provide recommendations to users about possible places to visit based on their past interests.

A very similar application is the WikiCity project [25] which collects real time information with the use of GPS and mobile devices. These are then used to collect the location patterns of users, and their use in a variety of neighborhoods.

8.1.3 MacroSense Application. The MacroSense application [195] is similar in terms of the data it collects and kind of functionality it provides; however it is focussed towards the commercial segment in predicting consumer behavior. The application can predict the behavior of customers based on their location profile and behavior. The application can predict what a particular customer may like next. The broad idea is to segment and cluster customers into marketing groups based on their behavior, and use this information in order to make predictions. For example, the popularity of a product with users who are most like the target can be used for predictive purposes. Thus, this approach is somewhat like collaborative filtering, except that it uses the behavior of customers rather than their feedback. The effectiveness of particular behaviors which predict the interests are also used. This analysis can be performed in real time, which provides great value in terms of predictive interactions. The analytics can also be used in order to predict group influences for the behaviors of the underlying subjects.

8.1.4 LiveCompare for Grocery Bargain Hunting. A system called *LiveCompare* [46] has been proposed for grocery bargain hunting with the use of participatory sensing. *LiveCompare* works with participating grocery store shoppers with camera-enabled mobile phones with internet access. Virtually, all smart phones today are camera- and internet-enabled, and therefore this requirement is quite a reasonable one, which does not require any additional expenditure from participating users. The camera phones are used in order to snap a picture of the product's price tag. We note that this price tag, typically contains a UPC bar code, from which information about the product can be ex-

tracted. The barcodes can be decoded from the photograph with the use of barcode libraries such as *ZXing*[193]. At this point, the numerical UPC value and the just-taken photograph are transferred to *LiveCompare*'s central server. This data is stored in *LiveCompare*'s database for use in future queries, and the UPC value determines the unique product for which price comparisons are requested. The client also sends its GPS or GSM cell information to the server so that the current store can be identified. This location information allows *LiveCompare* to limit query results to include only nearby stores. Results include store information and the option to view the time-stamped photographs associated with the specific product in question at each store. Users are not required to manually input pricing data in order to improve trustworthiness; this low burden of participation improves the ability to recruit participants during deployment. In any participatory system, it is recognized that to contribute data, users give up their time, attention, and mobile device's battery power. Therefore, it is critical to ensure that users have sufficient incentive to participate. *LiveCompare* directly addresses this challenge through its query protocol. When a user submits a query from a grocery store, he identifies the product for which he wants price comparison information by snapping a photograph of the product's price tag (including bar code). The server *appends the photograph submitted during the query to its database*. Thus, by requiring that a geotagged photograph be uploaded as part of a query, *LiveCompare* automatically populates its database whenever a user initiates a query. Thus, the principle of increasing incentive and participation is: "*To use, you must contribute.*"

The problem of sharing consumer prices with the use of mobile phones has started gaining attention recently. For example, the *Mobishop* system for sharing consumer prices with mobile phones has been proposed in [146]. Methods for sharing fuel prices with the help of a network of mobile phone cameras has been proposed in [50].

8.1.5 Location-Aware Search, Feedback, and Product Recommendation. Virtually all mobile phones have applications which enable GPS-based searches for popular businesses such as restaurants, coffee shops, gas stations, or department stores. For example, the *YellowPages* application on most mobile phones is now GPS enabled. Furthermore, many social review systems (which allow users to share their opinions about businesses) such as *Yelp* [199] integrate the social reviews with GPS-enabled search. This allows a user to not only search for business of interest, but even businesses which have positive reviews associated with them. These applications also allow users to

enter their feedback about their own experiences into the system. This unique combination of user-based text feedback and mobile sensing is powerful combination, which provides unprecedented information and flexibility in terms of combining location information with the social opinions of other users.

For shopping applications, the ability to perform *recommendations* is a useful functionality in a wide variety of scenarios. Since spatial location is highly corrected to user-buying behavior, it is natural to use GPS information for such applications. An important observation in this work is that some items or products (eg. restaurants) are spatial in nature, whereas others (eg. movies) are non-spatial in nature, since the user-experience with the product in not locality dependent. Similarly, ratings of a user may sometimes be spatial in nature, when some locations (eg. *FourSquare*) allow location-based check in and ratings. The work in [101], which uses location-based ratings for the recommendation process. The *LARS* [101] suppots a taxonomy of three classes of location-based ratings– (i) spatial ratings for non-spatial items, (ii) non-spatial ratings for spatial items, and (iii) spatial ratings for spatial items. *LARS* uses spatial partitioning in order to utilize spatially closer users for the recommendation process. This maximizes system scalability without affecting the recommendation quality. Furthermore, since users prefer closer locations for the purpose of their buying behavior, the spatial nature of items is used in order to recommend items which are closer to querying users. This is modeled with the use of a *travel penalty*. It has been shown in [101], that these features can be used either separately or together in order to maximize the effectiveness of the recommendation process.

8.1.6 Wikitude Augmented Reality Application. The wikitude application [191] is designed for mobile phones (such as *Blackberry* and *iPhone*, and uses the GPS location and the compass within mobile phones in order to provide an "augmented reality" experience from the mobile phone, by pointing it in different directions. The application is connected with social networking application such as *Facebook* and *Twitter*, and can collect messages, tweets and events from users within a particular neighborhood, and can be made available to the user. In addition, by pointing the device in a particular direction, it may be possible to find useful points of interest such as restaurants, shopping places, or movie theaters. It is even possible to determine mobile coupons and discounts from shops within a particular neighborhood with this kind of application.

8.1.7 Microsoft SensorMap. Most of the applications discussed above are based on location data, which is automatically collected based on user behavior. The SensorMap project [127] at Microsoft allows for a more general framework in which users can *choose to publish any kind of sensor data*, with the understanding that such shared knowledge can lead to interesting inferences from the data sets. For example, the sensor data published by a user could be their location information, audio or video feeds, or text which is typed on a keyboard. The goal of the SensorMap project is to store and index the data in a way such that it is efficiently searchable. The application also allows users to index and cache data, so that users can issue spatio-temporal queries on the shared data.

The SensorMap project is part of the SenseWeb project, which allows sharing and exploring of sensor streams over geo-centric interfaces. A number of key design challenges for managing such sensor streams have been discussed in [120]. Other key challenges, which are associated with issues such as the privacy issues involved with continuously collecting and using the sensors which are only intermittently available is discussed in [99].

8.2 RFID Technology: The Internet of Things

The general idea of social sensing can also be extended to applications which use RFID technology to track objects, as opposed to "social" sensing paradigms, which track people. This technology is also transformative for social sensing, because of the close relations between people and objects in many scenarios, and the social inferences, which may be possible with the use of such tracking technology. The idea is that radio frequency tags are attached to commercial products or other objects to be tracked, and these tags do little more than provide their unique *Electronic Product Code (EPC)* to nearby sensor readers. Thus, the movements of objects of interest can be identified by appropriate receivers at checkpoints where the object movement is tracked. Furthermore, these readers can be connected to the internet, where they can publish the data about the objects, and enable effective search, querying, and indexing of these objects with the use of the semantic web framework [82].

Animals, commercial products, baggage and other high volume items are often tracked with the use of Radio Frequency Identification (RFID) tags. For example, RFID technology has been used to track the movement of large animals such as whales with chips embedded in them. Such chips may sometimes even have transmitters embedded in them,

which can be picked up by satellite. RFID technology has even found application in a number of medical applications, in which RFID chips are embedded in patients in order to track their case history. RFID Technology has lead to the general vision of *the internet of things* [16], in which uniquely identifiable objects can be continuously tracked over time. In the case of commercial applications, the products may have implicit links among them which correspond to shared batches or processes during the production and transportation process. Such tracking data can be used in conjunction with linkage analysis in order to determine the causality and origin of tainted products. It can also be used to track the current location of other products which may be tainted. Such data is typically quite noisy, error-prone, incomplete, and massive in volume. Thus, this leads to numerous challenges in data compression, storage and querying. A detailed tutorial on RFID methods may be found in [81]. The technology is also discussed in some detail in a later chapter of this book [4, 5].

8.3 Vehicular Participatory Sensing

In vehicular participatory sensing, a variety of sensor data from vehicles such as mobile location, or other vehicular performance parameters may be continuously transmitted to users over time. Such data may be shared with other users *in the aggregate* in order to preserve privacy. This is the social aspect of such applications, since they enable useful individual decisions based on global patterns of behavior. In addition, vehicular participatory sensing may be used in order to enable quick responses in case of emergencies involving the vehicle operation. We note that much of the work discussed above for animal and moving object trajectory mining [104, 105, 112–115, 109, 110] are also applicable to the case of vehicular data. In addition, vehicular data poses unique challenges in terms of data collection, sensing, transmission and privacy issues. Classic examples of vehicular participatory sensing include the *CarTel* [88] and *GreenGPS* systems [64]. While we will focus on a detailed discussion of these systems as the most well known representatives of vehicular participatory sensing, a number of other sensing systems have been designed for different applications such as traffic monitoring and road conditions [124], cyclist experience mapping [55, 142], and the determination of transportation modes [144].

The problem of sharing bike track paths by different users has been explored in [142]. The problem of finding bike routes is naturally a trial-and-error process in terms of finding paths which are safe and enjoyable. The work in [142] designs *Biketastic*, which uses GPS-based sensing on

a mobile phone application in order to create a platform which enables rich sharing of biker experiences with one another. The microphone and the accelerometer embedded on the phone are sampled to infer route noise level and roughness. The speed can also be inferred directly from the GPS sensing abilities of the mobile phone. The platform combines this rich sensor data with mapping and visualization in order to provide an intuitive and visual interface for sharing information about the bike routes. A different application uses the time-stamped location information in order to determine the mobility profiles of individuals [144]. Next, we will discuss the *Cartel* and *GreenGPS* systems.

8.3.1 CarTel System. The *CarTel* project at MIT [88] is designed for mining and managing large amounts of sensor data, which are derived from vehicular participatory sensing. The most common data is vehicular position data, from which large amounts of information about road congestion, conditions, and other violations may be determined. The project focusses on the collection and use of such data in an efficient and privacy-preserving way. The actual data may be collected either from mobile phones in the car or from embedded devices within the car itself. For example, the Onboard Diagnostics Interface (OBD-II) equipped on modern cars can be used to collect tremendous amounts of useful data in this context. The OBD-II is a diagnostic system that monitors the health of the automobile using sensors that measure approximately 100 different engine parameters. Examples of monitored measurements include fuel consumption, engine RPM, coolant temperature and vehicle speed. Vehicles that have been sold in the United States after 1996 are mandatorily equipped with a sensing subsystem called the On-Board Diagnostic (OBD-II) system. A number of key components of the *CarTel* system are as follows:

Traffic Mitigation: In this case, two systems *VTrack* and *CTrack* [154, 155] have been proposed for processing error-prone position streams for estimating trajectory delays accurately. Since the location data is typically error-prone as a result of transmission errors, or outages, the technique is designed to be resistant to errors. In particular, the *CTrack* system [154] can work with the position data from cellular base stations, in which the location error is much higher than GPS data. The system continuously collects the data, and combines real-time and historic delay estimates to produce predictions of future delays at various points in time in the future. The results of the predictive model are sent to a commute portal where users can view the data along with appropriate traffic routing strategies.

Road Conditions: The idea in this approach [58] is to use the oppor-

tunistic mobility of sensor-equipped vehicles to detect and report the surface conditions of roads. Each car in the system carries 3-axis acceleration and GPS sensors, gathering location-tagged vibration data. The system uses uses *CarTel*'s opportunistic wireless protocols to deliver the data over whatever wireless network is available to a back-end server (discussed in detail below). The server processes this vibration data using machine learning techniques in order to predict the surface conditions.

Data Muling and Networking: The data collected in a vehicle (such as information about the road surface conditions) may sometimes need to be routed to a back-end server, even in cases where a continuous mobile connection is not available. In such cases, intermittent wifi access points may be available along the route of the vehicle. should use wireless networks opportunistically [57, 29]. The idea is to use a combination of WiFi, Bluetooth, and cellular connectivity, using whatever mode is available, while being completely transparent to underlying applications. In some cases, cars may be used as mules in order to carry the data, when direct connectivity is not available [29].

Query Processing of Intermittently Connected Data: Participatory sensing sensor network applications must cope with a combination of node mobility and high data rates when media-rich data such as audio, video or images are being captured by a sensors. As a result of the mobility, the sensor networks may display intermittent and variable network connectivity, and often have to deliver large quantities of data relative to the bandwidth available during periods of connectivity. In order to handle this challenge, a system known as *ICEDB (Intermittently Connected Embedded Database)* [178] was proposed, which incorporates a delay-tolerant continuous query processor, coordinated by a central server and distributed across the mobile nodes. The system contains algorithms for prioritizing certain query results to improve application-defined utility metrics.

Privacy Protection: The process of tracking the position of individual vehicles is fraught with numerous challenges from a privacy perspective. Therefore, techniques are needed to be able to compute appropriate functions on the location data, without violating individual privacy. The *CarTel* system provides excellent privacy protection of user location data, while being able to compute aggregate functions on the location statistics. This is called the *VPriv* system [134]. More details on this system are discussed in the section on privacy in this chapter.

8.3.2 Green GPS. Green GPS [64] is a participatory sensing navigation service that allows drivers to find the most fuel-efficient routes

for their vehicles between arbitrary end-points. Green GPS relies on data collected by individuals from their vehicles as well as on mathematical models to compute fuel efficient routes.

The most fuel efficient route may depend on the vehicle and may be different from the shortest or fastest route. For example, a fast route that uses a freeway may consume more fuel because fuel consumption increases non-linearly with speed. Similarly, the shortest route that traverses busy city streets may be suboptimal because of downtown traffic. The data collected by the different drivers can be used in conjunction with mathematical models in order to make effective predictions. A natural question arises as to the nature of the data which can be collected by the different individuals for this purpose. The service exploits measurements of standard vehicular sensor interfaces that give access to most gauges and engine instrumentation.

To build its fuel efficiency models, Green GPS utilizes a vehicle's OBD-II system and a typical scanner tool in conjunction with a participatory sensing framework. The team is collecting data from vehicles driven by research participants to determine what factors influence fuel consumption. The data collected by the participants is driving the creation of a mathematical model that enable computing fuel consumption of different cars on different road segments. Early studies have shown that a 13% reduction in consumer gas consumption is possible over the shortest path and 6% over the fastest path.

8.4 Participatory Sensing in Healthcare

A variety of *participatory sensing techniques* can be used for enabling real-time services. In participatory sensing, users agree to allow data about them to be transmitted in order to enable a variety of services which are enabled in real time. The ability to carry such devices allows its use for a variety of healthcare applications involving the elderly. For example, elderly patients can use this in order to call for care when necessary. Similarly, such sensing devices can be utilized for a variety of safety and health-care related applications.

Several companies such as *Vivometrics*, *Bodymedia*, and *Mini-mitter* have [196–198] have designed enhanced versions of the Holter ECG monitoring device [85], which is commonly used for ambulatory services. These enhanced devices are able to monitor a patient's ECG for longer periods of time, and transmit them remotely to the physician. Such a concept is very useful for high-risk populations (such as elderly patients), because it allows quick and time-critical responses, which has the potential to save lives. While *inpatient mobile sensing* is quite common in

medical domains, the advancement of this natural concept to more proactive applications such as round-the-clock monitoring has only been a recent development.

A method called *LiveNet* is proposed in [150], in which a flexible distributed mobile platform that can be deployed for a variety of proactive healthcare applications that can sense one's immediate context and provide feedback. This system is based on standard PDA hardware with customized sensors and a data acquisition hub, which provides the ability for local sensing, real-time processing, and distributed data streaming. This integrated monitoring system can also leverage off-body resources for wireless infrastructure, long-term data logging and storage, visualization/display, complex sensing, and computation-intensive processing. The *LiveNet* system also allows people to receive real-time feedback from their continuously monitored and analyzed health state. The system can also communicate health information to caregivers and other members of an individual's social network for support and interaction. One of the attractive features of this system is that it can combine general-purpose commodity hardware with specialized health/context sensing within a networked environment. This creates a multi-functional mobile healthcare device that is at the same time a personal real-time health monitor, which provides both feedback to the patient, the patient's social network, and health-care provider.

We note that a significant number of predictions can also be made without collecting data which is clinical in nature. In particular, the daily activities of an individual can provide key insights into their health. Smartphones have now become sophisticated enough that the data from the different sensors can be fused in order to infer the daily activities of an individual [65]. For example, the presence of illness and stress can affect individuals in terms of their total communication, interactions with respect to the time of day, the diversity and entropy of face-to-face communications and their movement. In order to achieve this goal, the work in [121] uses mobile phone based co-location and communication sensing to measure different attributes about the daily activity of an individual. It has been shown in [121], that the collection of even simple day-to-day information has a powerful effect on the ability to make an accurate diagnosis. Methods have also been proposed for finding sequential patterns from human activity streams, in order to determine the key activity trends over time. Furthermore, such activity monitoring cane be used to model the influence of different individuals on each other in terms of their daily activities. The work in [122] used a mobile phone platform to examine how individuals are influenced by face-to-face interactions in terms of their obesity, exercise and eating habits. It was shown that

such interactions do have a significant influence over individuals, which may propagate in the social network over time. Such an approach [54, 139, 140] has also been applied to the problem of geriatric care. This is because medical conditions such as dementia in older patients show up as specific kinds of activity patterns over time. It has been shown in [54], how such activity recognition methods can be used in the context of geriatric care.

From a predictive modeling perspective, a key challenge which arises is that a large amount of data may potentially need to be collected simultaneously from a large number of patients in order to make accurate real time predictions. This requires the design of fast data stream processing algorithms [7]. A recent paper [89] proposes a number of real-time data stream mining methods for fast and effective predictive modeling from sensor data. This kind of approach can be used for a wide variety of medical conditions, though the nature of the data collected and the predictive modeling would depend upon the nature of the disease modeling at hand. For example, the work in [84] discusses a variety of methods which can be used for diabetes monitoring with the use of collected data. Another interesting method for health and fitness monitoring has been developed in [119], in which modern mobile phones are used in order to both sense and classify the activities of an individual in real time. It has been shown that such machine learning algorithms can be used in conjunction with the collected data in order to provide effective monitoring and feedback. A discussion of some of the challenges in selecting sensors for health monitoring with the use of participatory sensing may be found in [41].

9. Future Challenges and Research Directions

In this chapter, we examined the emerging area of integrating sensors and social networks. Such applications have become more commonplace in recent years because of new technologies which allow the embedding of small and unobtrusive sensors in clothing. The main challenges of using such technologies are as follows:

- Such applications are often implemented on a very large scale. In such cases, the database scalability issues continue to be a challenge. While new advances in stream processing have encouraged the development of effective techniques for data compression and mining, mobile applications continue to be a challenge because of the fact that *both* the *number* of streams and rate of data collection may be extremely large.

- A major challenge in sensor-based social networking are the privacy issues inherent in the underlying applications. For example, individuals may not be willing to disclose their locations [66] in order to enable applications such as proximity alerts. In many cases, such constraints can greatly reduce the functionality of such applications. A major challenge in such applications is to provide individual hard guarantees on their privacy level, so that they become more willing to share their real time information.

- The trust issues continue to be a challenge for such applications, because of the openness of such systems in allowing participants to contribute information. Furthermore, the goals of privacy and trust seem to be at odds with one another, because the former is achieved by lowering data fidelity, and the latter requires higher data fidelity.

- Battery life continues to be a severe constraint in such applications. Therefore, it is critical to tailor application design to work efficiently in power-constrained scenarios.

- The architectural challenges for such systems continue to be quite extensive. For example, the use of centralized processing methods for such large systems does not scale well. Therefore, new methods [120, 127] have moved away from the centralized architecture for stream collection and processing.

The future challenges of such research include the development of new algorithms for large scale data collection, processing and storage. Some advancements [7, 120, 127] have already been made in this direction.

Acknowledgements

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, J. Reich. Mobiscopes for Human Spaces. *IEEE Pervasive*, 6(2), pp. 20–29, April 2007.

[2] O. Abul, F. Bonchi, M. Nanni. Never Walk Alone: Uncertainty for Anonymity in Moving Object Databases, *ICDE Conference*, 2008.

[3] H. Ahmadi, N. Pham, R. Ganti, T. Abdelzaher, S. Nath, J. Han. Privacy-aware regression modeling of participatory sensing data. *SenSys*, 2010.

[4] C. C. Aggarwal, J. Han. A Survey of RFID Data Processing, *Managing and Mining Sensor Data*, Springer, 2013.

[5] C. C. Aggarwal, N. Ashish, A. Sheth. The Internet of Things: A Survey from the Data-Centric Perspective, *Managing and Mining Sensor Data*, Springer, 2013.

[6] C. C. Aggarwal, T. Abdelzaher. Integrating Sensors and Social Networks, *Social Network Data Analytics*, Springer, 2011.

[7] C. C. Aggarwal (ed.) Data Streams: Models and Algorithms, *Springer*, 2007.

[8] C. C. Aggarwal, H. Wang (ed.) Managing and Mining Graph Data, *Springer*, 2010.

[9] C. C Aggarwal, P. Yu (ed.) Privacy-Preserving Data Mining: Models and Algorithms, *Springer*, 2008.

[10] C. C. Aggarwal, P. S. Yu. Online Analysis of Community Evolution in Data Streams, *SIAM Conference on Data Mining*, 2005.

[11] C. C. Aggarwal, S. Lin, P. Yu. On Influential Node Discovery in Dynamic Social Networks, *SDM Conference*, 2012.

[12] C. C. Aggarwal, Y. Zhao, P. Yu. On Clustering Graph streams, *SIAM Conference on Data Mining*, 2010.

[13] D. Agrawal, C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD Symposium on Principles of Database Systems*, pages 247–255, 2001.

[14] R. Agrawal, R. Srikant. Privacy preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, TX, May 2000.

[15] C. Asavathiratham, The lnfluence Model: A Tractable Representation for the Dynamics of Networked Markov Chains, *TR, Dept. of EECS, MIT*, Cambridge, 2000.

[16] K. Ashton. That 'Internet of Things' Thing. In: *RFID Journal*, 22 July, 2009.

[17] B. Bamba, L. Liu, P. Pesti, T. Wang. Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid, *World Wide Web Conference*, 2008.

[18] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, M. Corner. Virtual Compass: Relative Positioning to Sense Mobile Social Interactions. *Pervasive*, 2010.

[19] J. Bao, M. Mokbel, C.-Y. Chow. GeoFeed: A Location Aware News Feed System. *ICDE Conference*, pp. 54–65, 2012.

[20] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Misra, K. Seada. Fusing Mobile, Sensor and Social Data to Fully Enable Context Aware Computing, *Hotmobile*, 2010.

[21] A. Beberg, V. S. Pande. Folding@home: lessons from eight years of distributed computing. *IEEE International Parallel and Distributed Processing Symposium*, pp. 1–8, 2009

[22] M. Benkert, J. Gudmundsson, F. Hubner, T. Wolle. Reporting flock patterns. *COMGEO*, 2008.

[23] D. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. *The Journal of Research into New Media Technologies*, 14(1), pp. 75-90, 2008.

[24] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. B. Srivastava. Participatory Sensing, *WWW Conference*, 2006.

[25] F. Calabrese, K. Kloeckl, C. Ratti. Wikicity: Real-Time Urban Environments. *IEEE Pervasive Computing*, 6(3), 52-53, 2007. `http://senseable.mit.edu/wikicity/rome/`

[26] A. T. Campbell et al. The Rise of People Centric Sensing, *IEEE Internet Computing*, 12(4), 2008.

[27] X. Cao, G. Cong, C. S. Jensen. Mining Significant Semantic Locations From GPS Data. *VLDB Conference*, 2010.

[28] D. Chakrabarti, R. Kumar, A. Tomkins. Evolutionary clustering. *KDD Conference*, 2006.

[29] K. Chen. CafNet: A Carry-and-Forward Delay-Tolerant Network. *MEng Thesis, MIT EECS*, Feb. 2007.

[30] R. Cheng, Y. Zhang, E. Bertino, S. Prabhakar. Preserving user location privacy in mobile data management infrastructures. *Privacy Enhancing Technologies*, 2006.

[31] Y. Chi, X. Song, D. Zhou, K. Hino, B. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. *ACM KDD Conference*, 2007.

[32] M. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, C. Yu. Automatic construction of travel itineraries using social breadcrumbs. *HT*, 2010.

[33] T. Choudhury A. Pentland. The Sociometer: A Wearable Device for Understanding Human Networks. *International Sunbelt Social Network Conference*, February 2003.

[34] T. Choudhury, A. Pentland. Sensing and Modeling Human Networks using the Sociometer, *International Conference on Wearable Computing*, 2003.

[35] T. Choudhury, A. Pentland. Characterizing Social Networks using the Sociometer. *North American Association of Computational Social and Organizational Science*, 2004.

[36] T. Choudhury, B. Clarkson, S. Basu, A. Pentland. Learning Communities: Connectivity and Dynamics of Interacting Agents. *International Joint Conference on Neural Networks*, 2003.

[37] T. Choudhury, M. Philipose, D. Wyatt, J. Lester. Towards Activity Databases: Using Sensors and Statistical Models to Summarize People's Lives. *IEEE Data Engineering Bulletin*, Vol. 29 No. 1, March 2006.

[38] C.-Y. Chow, M. F. Mokbel. Privacy of Spatial Trajectories. *Computing with Spatial Trajectories*, pp. 109–141, 2011.

[39] A. Clauset, M. E. J. Newman, C. Moore. Finding community structure in very large networks. *Phys. Rev. E 70, 066111*, 2004.

[40] I. Constandache, R. Choudhury, I. Rhee. Towards mobile phone localization without war-driving, *INFOCOM Conference*, 2010.

[41] D. Cook, L. Holder. Sensor selection to support practical use of health-monitoring smart environments. *Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery* 1(4): pp. 339–351, 2011.

[42] D. Cook, W. Song. Ambient Intelligence and Wearable Computing: Sensors on the Body, in the Home and Beyond. *JAISE*, 1(2): pp. 83–86, 2009.

[43] C. Costa, C. Laoudias, D. Zeinalipour-Yazti, D. Gunopulos. SmartTrace: Finding similar trajectories in smartphone networks without disclosing the traces. *ICDE Conference*, 2011.

[44] M. Coscia, S. Rinzivillo, F. Giannotti, D. Pedreschi. Optimal Spatial Resolution for the Analysis of Human Mobility, *ASONAM Conference*, 2012.

[45] L. P. Cox. Truth in Crowdsourcing, *IEEE Journal on Security and Privacy*, Volume 9, Issue 5, September 2011.

[46] L. Deng, L. P. Cox. Livecompare: grocery bargain hunting through participatory sensing. *HotMobile*, 2009.

[47] A. Doan, R. Ramakrishnan, A. Halevy. Crowdsourcing systems on the World-Wide Web. *Communications of the ACM*, 54(4), pp. 86–96, 2011.

[48] X. Dong, L. Berti-Equille, D. Srivastava. Truth discovery and copying detection in a dynamic world. *VLDB*, 2(1): pp. 562–573, 2009.

[49] X. Dong, L. Berti-Equille, D. Srivastava. Integrating conflicting data: the role of source dependence. *Proc. VLDB Endow.*, 2: pp. 550–561, August 2009.

[50] Y. Dong, S.S. Kanhere, C.T. Chou, N. Bulusu. Automatic collection of fuel prices from a network of mobile cameras, *IEEE DCOSS*, 2008.

[51] A. Dua, N. Bulusu, W. Feng, W. Hu. Towards Trustworthy Participatory Sensing. *Proceedings of the Usenix Workshop on Hot Topics in Security*, 2009.

[52] N. Eagle, A. Pentland. Reality Mining: Sensing Complex Social Systems, *Personal and Ubiquitous Computing*, 10(4), 2006.

[53] N. Eagle, A. Pentland, D. Lazer. Inferring social network structure using mobile phone data. *Proceedings of the National Academy of Sciences (PNAS)*, 106:15274–15278, 2009.

[54] M. Schmitter-Edgecombe, P. Rashidi, D. Cook, L. Holder. Discovering and Tracking Activities for Assisted Living, *The American Journal of Geriatric Psychiatry*, In Press, 2011.

[55] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, A. Campbell. The Bikenet Mobile Sensing System for cyclist experience mapping, *ACM Sensys*, 2007. Extended version appears in *ACM TOSN*, 6(1), 2009.

[56] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *OSDI*, 2010.

[57] J. Eriksson, H. Balakrishnan, S. Madden, Cabernet: Vehicular Content Delivery Using WiFi. *ACM MOBICOM Conference*, 2008.

[58] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, H. Balakrishnan. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. *MobiSys*, 2008.

[59] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the SIGMOD/PODS Conference*, pages 211–222, 2003.

[60] C. Faloutsos, T. Kolda, J. Sun. Mining Large Time-Evolving Data using Matrix and Tensor Tools, *ICDM Conference*, 2007.

[61] R. K. Ganti, N. Pham, Y.-E. Tsai, T. F. Abdelzaher. PoolView: Stream Privacy in Grassroots Participatory Sensing, *SenSys*, 2008.

[62] R. K. Ganti, Y.-E. Tsai, T. F. Abdelzaher. SenseWorld: Towards Cyber-Physical Social Networks. *IPSN*, pp. 563–564, 2008.

[63] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, J. A. Stankovic, SATIRE: A Software Architecture for Smart AtTIRE. *Mobisys*, 2006.

[64] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, T. Abdelzaher. GreenGPS: A Participatory Sensing Fuel-Efficient Maps Application. *Mobisys*, San Francisco, CA, June 2010.

[65] R. K. Ganti, S. Srinivasan, A. Gacic. Multi-sensor fusion in smartphones for lifestyle monitoring, *International Conference on Body Sensor Networks*, 2010.

[66] B. Gedik, L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. *ICDCS Conference*, 2005.

[67] G. Ghinita, P. Kalnis, S. Skiadopoulos. PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems. *WWW Conference*, 2007.

[68] G. Ghinita, M. Damiani, C. Silvestri, E. Bertino. Preventing velocity-based linkage attacks in location-aware applications. *GIS*, 2009.

[69] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, K.-L. Tan. Private queries in location based services: anonymizers are not necessary. *SIGMOD Conference*, 2008

[70] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi. Trajectory pattern mining. *ACM KDD Conference*, 2007.

[71] P. Gilbert, L. Cox, J. Jung, D. Wetherall. Towards Trustworthy Mobile Sensing, *Hotmobile*, 2010.

[72] P. Gilbert, J. Jung, K. Lee, H. Qin, D. Sharkey, A. Sheth, L. Cox. YouProve: Authenticity and Fidelity in Mobile Sensing, *ACM Sensys*, 2011.

[73] M. Gonzlez, Hidalgo, A.-L. Barabasi. Understanding Individual Human Mobility Patterns, *Nature*, 453: pp. 779–782, 2008.

[74] M. Gotz, S. Nath, J. Gehrke. MaskIt: Privately releasing user context streams for personalized mobile applications. *ACM SIGMOD Conference*, 2012.

[75] M. Gruteser, D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. *MobiSys*, 2003.

[76] M. Gruteser, X. Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security and Privacy*, 2: pp. 28–34, 2004.

[77] J. Gudmundsson, M. van Kreveld. Computing longest duration flocks in trajectory data. *GIS*, 2006.

[78] T. Guo, K. Iwamura, M. Koga. Towards high accuracy road maps generation from massive GPS traces data. *Proc. of IGARSS*, pp. 667–670, 2007.

[79] M. Gupta, C. Aggarwal, J. Han, Y. Sun. Evolutionary Clustering and Analysis of Heterogeneous Bibliographic Networks, *ASONAM Conference*, 2011.

[80] R. Guting, M. Schneider. Moving Objects Databases, *Morgan Kaufmann*, 2005.

[81] J. Han, J.-G. Lee, H. Gonzalez, X. Li. Mining Massive RFID, Trajectory, and Traffic Data Sets (Tutorial). *ACM KDD Conference*, 2008.
Video of Tutoral Lecture at: `http://videolectures.net/kdd08_han_mmrfid/`

[82] O. Hassanzadeh, A. Kementsietsidis. Data Management Issues for the Semantic Web, *ICDE Conference*, 2012.

[83] Y. He, S. Barman, D. Wang, J. Naughton. On the complexity of privacy-preserving complex event processing. *PODS*, 2011.

[84] A. Helal, D. Cook, M. Schmatz. Smart Home-Based Health Platform for Behavioral Monitoring and Alteration of Diabetes Patients. *Journal of Diabetes Science and Technology*, 3(1): pp. 141–148, January 2009.

[85] N. J. Holter, J. A. Generelli. Remote recording of physiologic data by radio. *Rocky Mountain Medical Journal*, pp. 747–751, 1949.

[86] Z. Huang, W. Du, B. Chen. Deriving private information from randomized data. In *Proceedings of the 2005 ACM SIGMOD Conference*, pages 37–48, Baltimore, MD, June 2005.

[87] K. Huang, S. Kanhere, W. Hu. Preserving Privacy in Participatory Sensing Systems, *Computer Communications*, 33, pp. 1266–1280, 2010.

[88] B. Hull, V. Bychkovsky, K. Chen, M. Goraczko, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, S. Madden, CarTel: A Distributed Mobile Sensor Computing System, *ACM SenSys*, 2006.

[89] V. Jakkula, D. Cook, G. Jain. Prediction Models for a Smart Home based Health Care System. *Advanced Information Networking and Applications Workshops*, 2007.

[90] H. Jeung, H. T. Shen, X. Zhou. Convoy queries in spatio-temporal databases. *ICDE Conference*, 2008.

[91] H. Jeung, M. L. Yiu, X. Zhou, C. Jensen, H. Shen, Discovery of Convoys in Trajectory Databases, *VLDB Conference*, 2008.

[92] Y. Jing, S. Baluja. Pagerank for product image search. *WWW Conference*, pages 307–316, 2008.

[93] P. Kalnis, N. Mamoulis, S. Bakiras. On discovering moving clusters in spatio-temporal data. *SSTD*, 2005.

[94] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, D. Kotz. AnonySense: opportunistic and privacy-preserving context collection, *Proceedings of Sixth International Conference on Pervasive Computing (Pervasive)*, 2007.

[95] H. Kargupta, S. Datta, Q. Wang, K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the IEEE International Conference on Data Mining*, pages 99–106, 2003.

[96] D. Kempe, J. Kleinberg, E. Tardos, Maximizing the Spread of Influence in a Social Network, *ACM KDD Conference*, 2003.

[97] D. Kim, Y. Kim, D. Estrin, M. B. Srivastava. Sensloc: Sensing everyday places and paths using less energy, *ACM Sensys Conference*, 2010.

[98] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins. Trawling the web for emerging cyber-communities. *WWW Conference*, 1999.

[99] A. Krause, E. Horvitz, A. Kansal, F. Zhao. Toward Community Sensing. *IPSN*, pp. 481–492, 2008.

[100] M. Laibowitz, N.-W. Gong, J. A. Paradiso, Wearable Sensing for Dynamic Management of Dense Ubiquitous Media. *BSN*, 2009.

[101] J. Levandoski, M. Sarwat, A. Eldawy, M. Mokbel. LARS: A Location-Aware Recommender System. *ICDE Conference*, pp. 450–461, 2012.

[102] P. Laube, S. Imfeld. Analyzing relative motion within groups of trackable moving point objects. *GIS*, 2002.

[103] H. K. Le, J. Pasternack, H. Ahmadi, M. Gupta, Y. Sun, T. Abdelzaher, J. Han, D. Roth, B. Szymanski, S. Adali. Apollo: Towards factfinding in participatory sensing. *IPSN Conference*, 2011.

[104] J.-G. Lee, J. Han, K.-Y. Whang, Trajectory Clustering: A Partition-and-Group Framework, *ACM SIGMOD Conference*, 2007.

[105] J.-G. Lee, J. Han, X. Li. Trajectory Outlier Detection: A Partition-and-Detect Framework, *ICDE Conference*, 2008.

[106] J.-G. Lee, J. Han, X. Li, H. Gonzalez. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB*, 1(1): pp. 1081–1094, 2008.

[107] V. Lenders, E. Koukoumidis, P. Zhang, M. Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. *Hotmobile*, 2008.

[108] J. Leskovec, K. J. Lang, A. Dasgupta, M. W. Mahoney. Statistical properties of community structure in large social and information networks. *WWW Conference*, 2008.

[109] X. Li, Z. Li, J. Han, J.-G. Lee. Temporal Outlier Detection in Vehicle Traffic Data. *ICDE Conference*, 2009.

[110] X. Li, J. Han, S. Kim, H. Gonzalez. ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets. *SDM Conference*, 2007.

[111] X. Li, J. Han, J.-G. Lee, H. Gonzalez. Traffic Density-Based Discovery of Hot Routes in Road Networks. *SSTD*, 2007.

[112] Y. Li, J. Han, J. Yang. Clustering Moving Objects, *ACM KDD Conference*, 2004.

[113] Z. Li, B. Ding, J. Han, R. Kays. Swarm: Mining Relaxed Temporal Moving Object Clusters. *VLDB Conference*, 2010.

[114] Z. Li, B. Ding, J. Han, R. Kays, Mining Hidden Periodic Behaviors for Moving Objects, *ACM KDD Conference*, 2010.

[115] Z. Li, J. Han, M. Ji, L. A. Tang, Y. Yu, B. Ding, J.-G. Lee, R. Kays. MoveMine: Mining moving object data for discovery of animal movement patterns. *ACM TIST*, 2(4): 37, 2011.

[116] Z. Li, C. X. Lin, B. Ding, J. Han. Mining Significant Time Intervals for Relationship Detection. *SSTD Conference*, 2011.

[117] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, J. A. Paradiso. A platform for ubiquitous sensor deployment in occupational and domestic environments. *IPSN*, 2007.

[118] Z. Liu, J. Xu Yu, Y. Ke, X. Lin, L. Chen. Spotting Significant Changing Subgraphs in Evolving Graphs, *ICDM Conference*, 2008.

[119] B. Longstaff, S. Reddy, D. Estrin. Improving activity classification for health applications on mobile devices using active and semi-supervised learning, *ICST Conference on Pervasive Computing Technologies for Healthcare* , 2010.

[120] L. Luo, A. Kansal, S. Nath, F. Zhao. Sharing and exploring sensor streams over geocentric interfaces, *ACM SIGSPATIAL international conference on Advances in geographic information systems*, 2008.

[121] A. Madan, D. Cebrian, D. Lazer, A. Pentland. Social Sensing for Epidemiological Behavior Change, *Ubicomp*, 2010.

[122] A. Madan, S. Moturu, D. Lazer, A. Pentland. Social Sensing: Obsesity, Healthy Eating and Exercise in Face-to-Face Networks, *Wireless Health*, 2010.

[123] E. Miluzzo, N. Lane, K. Fodor, R. Peterson, S. Eisenman, H. Lu, M. Musolesi, X. Zheng, A. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe Application, *SenSys*, 2008.

[124] P. Mohan, V. Padmanabhan, R. Ramjee, Nericell: rich monitoring of road and traffic conditions using mobile smartphones, *ACM SenSys*, 2008.

[125] M. Mokbel, C. Chow, W. G. Aref. The New Casper: Query Processing for Location-based Services without Compromising Privacy, *VLDB Conference*, 2006.

[126] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh. Location privacy via private proximity testing. *NDSS*, 2011.

[127] S. Nath, J. Liu, F, Zhao. SensorMap for Wide-Area Sensor Webs. *IEEE Computer*, 40(7): 90-93, 2008.

[128] M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 2006.

[129] J. Paek, J. Kim, R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones, *MobiSys*, 2010.

[130] S. Papadimitrioum F. Li, G. Kollios, P. Yu. Time series compressibility and privacy. In *VLDB Conference*, 2007.

[131] L. Pareschi, D. Riboni, A. Agostini, C. Bettini. Composition and generalization of context data for privacy preservation. *PerComm*, 2008.

[132] N. Pham, T. Abdelzaher, S. Nath. On Bounding Data Stream Privacy in Distributed Cyber-physical Systems, *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (IEEE SUTC)*, Newport Beach, CA, June, 2010.

[133] N. Pham, R. K. Ganti, Y. S. Uddin, S. Nath, T. Abdelzaher, Privacy preserving Reconstruction of Multidimensional Data Maps in Vehicular Participatory Sensing, *EWSN*, 2010.

[134] R. A. Popa, H. Balakrishnan, A. Blumberg. VPriv: Protecting Privacy in Location-Based Vehicular Services. *USENIX Security Symposium*, 2008.

[135] B. Priyantha, D. Lymberopoulos, J. Liu. Enabling energy-efficient continuous sensing on mobile phones with Littlerock, *Information Processing in Sensor Networks*, 2010.

[136] G. Qi, C. Aggarwal, T. Huang. Community Detection with Edge Content in Social Media Networks, *ICDE Conference*, 2012.

[137] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, M. J. Neely. Energy-delay tradeoffs in smartphone applications, *MobiSys*, 2010.

[138] S. Ramchurn, D. Huynh, N. Jennings. Trust in Multi-agent Systems, *Knowledge Engineering Systems*, 19(1), pp. 1–25, 2004.

[139] P. Rashidi, D. Cook. Mining Sensor Streams for Discovering Human Activity Patterns over Time. *ICDM Conference*, 2010.

[140] P. Rashidi, D. Cook, L. Holder, M. Schmitter-Edgecombe. Discovering Activities to Recognize and Track in a Smart Environment, *IEEE TKDE*, 23(4), pp. 527–539, 2011.

[141] V. Rastogi, S. Nath. Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption, *ACM SIGMOD Conference*, 2010.

[142] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, M. B. Srivastava. Biketastic: sensing and mapping for better biking. *CHI*, pp. 1817–1820, 2010.

[143] S. Reddy, D. Estrin, M. B. Srivastava. Recruitment Framework for Participatory Sensing Data Collections.*Pervasive*, pp. 138–155, 2010.

[144] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, M. Srivastava. Using mobile phones to determine transportation modes, *ACM Transactions on Sensor Networks*, 6(3), pp. 1–27, 2010.

[145] M. Romaine, J. Richardson. State of the Translation Industry. *Translation Industry Report*, My Gengo, 2009.

[146] S. Sehgal, S.S. Kanhere, C.T. Chou. Mobishop: using mobile phones for sharing consumer pricing information, (Demo paper), *IEEE DCOSS*, 2008.

[147] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, P. Kalnis. User Oriented Trajectory Search for Trip Recommendation, *EDBT Conference*, 2012.

[148] K. Shilton, D. Estrin, R. Govindan, J. Kang. Designing the Personal Data Stream: Enabling Participatory Privacy in Mobile Personal Sensing. In *Research Conference on Comunication, Information, and Internet Policy (TPRC)*, 2009.

[149] L. Stenneth, P. S. Yu, O. Wolfson. Mobile Systems Location Privacy: "MobiPriv" A Robust k-anonymous System. *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2010.

[150] M. Sung, C. Marci, A. Pentland. Wearable Feedback Systems for Rehabilitation, *Journal of Neuroengineering and Rehabilitation*, 2:17, 2005.

[151] J. Sun, S. Papadimitriou, P. Yu, C. Faloutsos. Graphscope: Parameter-free Mining of Large Time-Evolving Graphs, *KDD Conference*, 2007.

[152] K.P. Tang, J. Fogarty, P. Keyani, J.I. Hong. Putting people in their place: an anonymous and privacy-sensitive approach to collecting sensed data in location-based applications, *SIGCHI*, 2006.

[153] L. Tang, X. Yu, S. Kim, J. Han, C. Hung, W Peng. Tru-Alarm: Trustworthiness Analysis of Sensor Networks in Cyber-Physical System. *ICDM Conference*, 2010.

[154] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, L. Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. *Proceedings of the NSDI*, 2011.

[155] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, H. Balakrishnan. VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones. *ACM SenSys*, 2009.

[156] H. Tong, S. Papadimitriou, P. Yu, C. Faloutsos. Proximity-Tracking on Time-Evolving Bipartite Graphs, *SDM Conference*, 2008.

[157] J. Tsai, P. Kelley, L. Cranor, N. Sadeh. Location sharing technologies: Privacy risks and controls. I/S: A *Journal of Law and Policy for the Information Society*, 6(2), 2010.

[158] Z. Yang, S. Zhong, R. N. Wright. Privacy-preserving classification without loss of accuracy. In *Proceedings of the Fifth SIAM International Conference on Data Mining*, pages 92–102, 2005.

[159] D. Wang, T, Abdelzaher, H. Ahmadi, J. Pasternack, D. Roth, M. Gupta, J. Han, O. Fatemieh, H. Le, C. Aggarwal. On Bayesian Interpretation of Fact-finding in Information Networks. *Fusion*, 2011.

[160] D. Wang, T. Abdelzaher, L. Kaplan, C. Aggarwal. On Quantifying the Accuracy of Maximum Likelihood Estimation of Participant Reliability in Social Sensing. *DMSN*, 2011.

[161] D. Wang, L. Kaplan, H. Le, T. F. Abdelzaher. On truth discovery in social sensing: a maximum likelihood estimation approach. *IPSN Conference*, 2012.

[162] D. Wang, T. Abdelzaher, L. Kaplan, C. Aggarwal. On Scalability and Robustness Limitations of Real and Asymptotic Confidence Bounds in Social Sensing, *SECON Conference*, 2012.

[163] M. Wirz, D. Roggen, G. Troster, Decentralized detection of group formations from wearable acceleration sensors, in *Proceedings IEEE SocialCom*, 2009

[164] D. Wyatt, T. Choudhury, J. Bilmes. Creating Social Network Models from Sensor Data, *NIPS Network Workshop*, 2007.

[165] D. Wyatt, T. Choudhury, J. Bilmes. Conversation Detection and Speaker Segmentation in Privacy Sensitive Situated Speech Data. *Proceedings of Interspeech*, 2007.

[166] D. Wyatt, T. Choudhury, H. Kautz. Capturing Spontaneous Conversation and Social Dynamics: A Privacy-Sensitive Data Collection Effort. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.

[167] D. Wyatt, T. Choudhury, J. Bilmes. Learning Hidden Curved Exponential Random Graph Models to Infer Face-to-Face Interaction Networks from Situated Speech Data. *Proceedings of AAAI*, 2008.

[168] T. Yan, V. Kumar, D. Ganesan. Crowdsearch: Exploiting crowds for accurate real-time image search on mobile phones, *MobiSys*, 2010.

[169] H. Yang, S. Chen, M. Lyu, I. King. Location-based Topic Evolution, *1st International Workshop on Mobile Location-based Services*, 2011. `http://dl.acm.org/citation.cfm?id=2025894`

[170] X. Yin, J. Han, P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE TKDE*, 2008.

[171] Z. Yin, L. Cao, J. Han, J. Luo, T. Huang. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. *SDM Conference*, 2011.

[172] Z. Yin, L. Cao, J. Han, C. Zhai, T. Huang. Geographical topic discovery and comparison. *WWW Conference*, 2011.

[173] X. Yu, A. Pan, L. A. Tang, Z. Li, J. Han. Geo-Friends Recommendation in GPS-based Cyber-physical Social Network. *ASONAM Conference*, 2011.

[174] D. Zeinalipour-Yazti, C. Laoudias, M. Andreou, D. Gunopulos. Disclosure-Free GPS Trace Search in Smartphone Networks. *Mobile Data Management*, 2011.

[175] T. Zhang, A. Popescul, B. Dom. Linear prediction models with graph regularization for web-page categorization. In *KDD*, pages 821–826, 2006.

[176] N. Zhang, S. Wang, W. Zhao. A new scheme on privacy-preserving data classification. In *ACM KDD Conference*, pages 374–383, 2005.

[177] F. Zhao, L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann, 2004.

[178] Y. Zhang, B. Hull, H. Balakrishnan, S. Madden. IceDB: Continuous Query Processing in an Intermittently Connected World. *ICDE Conference*, 2007.

[179] V. W. Zheng, Y. Zheng, X. Xie, Q. Yang. Collaborative location and activity recommendations with gps history data. *WWW Conference*, 2010.

[180] Y. Zheng, L. Liu, L. Wang, X. Xie. Learning transportation mode from raw gps data for geographic applications on the web, *WWW Conference*, 2008.

[181] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. *WWW Conference*, 2009.

[182] Y. Zhou, H. Cheng, J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1): pp. 718–729, 2009.

[183] Z. Zhuang, K.-H. Kim, J. P. Singh. Improving energy efficiency of location sensing on smartphones, *MobiSys*, 2010.

[184] `http://latitude.google.com`

[185] `http://www.citysense.com`

[186] `http://www.movebank.org`

[187] `http://www-01.ibm.com/software/data/infosphere/streams/`

[188] `http://www.navizon.com`

[189] `http://ilocalis.com`

[190] `http://www.trapster.com`

[191] `http;//www.wikitude.com`

[192] https://www.mturk.com/mturk/welcome

[193] http://code.google.com/p/zxing

[194] http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/

[195] https://www.sensenetworks.com/products/macrosense-technology-platform/

[196] Vivometrics, http://www.vivometrics.com

[197] SenseWear, BodyMedia Corporation http://www.bodymedia.com

[198] Mini Logger, Minimitter Corporation http://www.minimitter.com

[199] Yelp Business Review Site http://www.yelp.com

Chapter 10

SENSING FOR MOBILE OBJECTS

Nicholas D. Larusso
UC Santa Barbara
Dept. of Computer Science
nlarusso@cs.ucsb.edu

Ambuj K. Singh
UC Santa Barbara
Dept. of Computer Science
ambuj@cs.ucsb.edu

Abstract Recent advances in affordable positioning hardware and software have made the availability of location data ubiquitous. Personal devices such as tablet PCs, smart phones and even sport watches are all able to collect and store a user's location over time, providing an ever-growing supply of spatiotemporal data. Managing this plethora of data is a relatively new challenge and there has been a great deal of research in the recent years devoted to the problems that arise from spatiotemporal data. This book chapter surveys recent developments in the techniques used for the management and mining of spatiotemporal data. We focus our survey on three main areas: (i) *data management*, which includes indexing and querying mobile objects, (ii) *tracking*, making use of noisy location observations to infer an object's actual or future position, and (iii) *mining*, extracting interesting patterns from spatiotemporal data. First, we cover recent advances in database systems for managing spatiotemporal data, including index structures and efficient algorithms for processing queries. Next, we review the problem of tracking for mobile objects to estimate an object's location given a sequence of noisy observations. We discuss some of the common approaches used for tracking and examine some recent work which focuses specifically on tracking vehicles using a road network. Then we review the recent literature on mining spatiotemporal data. We conclude by discussing some interesting areas of future research.

Keywords: Mobility, Spatiotemporal Data Management, Querying Mobile Objects, Movement Modeling,

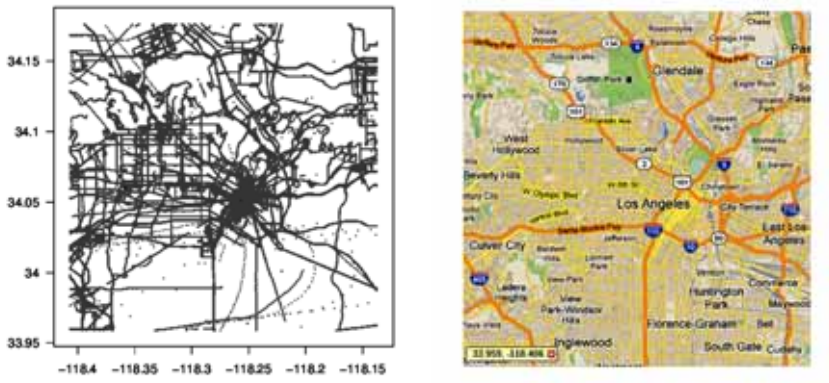
1. Introduction

The use of sensors that capture user location information over time is rapidly expanding and the various types of sensors are becoming ubiquitous. Location sensors are integrated into a large number of personal devices including PDAs, smartphones, and watches [104, 105]. Among the sensors used to acquire spatiotemporal data, the most popular is the Global Positioning System (GPS). GPS receivers are commonly embedded into vehicles for trip navigation, sports watches to track and monitor personal progress in hiking and running, and smart phones to provide general purpose location aware querying. The increased availability of positioning data has given rise to location based services (LBS), which utilizes a user's current position in order to personalize results. For instance, suppose a user searches for coffee shops on her smartphone. The query results will be filtered using both the relevancy from the search string as well as her position to return popular coffee shops which are physically close.

As an example of the availability of location data, figure 1 shows GPS data uploaded to [106] by users in a region of Los Angeles, CA along with a comparison to the underlying road network of the same area. From the figure, we can see two things: first, there is a large amount of spatiotemporal data available online. Second, the data is generated over several modes of transportation. Given how well the GPS trajectories outline the heavily traveled roads, it is clear that most of the data is collected while users are in vehicles. However, upon closer inspection, we can see that some trajectories cannot possibly be associated with an automobile and therefore must have been collected using some other mode of transportation (i.e. train, subway, or walking).

The adoption of new LBS has increased nearly as quickly as the technologies have become available. As of 2011, approximately 75% of smartphone users are using their devices for navigational purposes (e.g. driving directions) and 95% use their smart phones for location based searching [104, 105]. Coupled with the increasing number of smartphones year after year (approximately a 13% increase from 2010 to 2011 [104]), this signals a steep upward trend for LBS.

Given the availability of large quantities of spatiotemporal data, it is possible to ask several interesting questions about object movement. For example, assume we have a database containing the current positions of a



(a) GPS locations from trajectories (b) Corresponding map of Los Angeles, CA

Figure 10.1. Figure (a) shows a scatter plot of GPS points of approximately 200 GPS trajectories. Figure (b) is a road map of the underlying region of Los Angeles, CA, from which these points were collected. In addition to providing an outline of the highly traveled roads in LA, it can be seen that the GPS data represents multiple modes of transportation (i.e. automobile, train, and walking).

fleet of taxis picking up and dropping off customers throughout the city. There are several interesting queries and data management problems in this setting. First, given the current position of each taxi as well as the state (occupied or unoccupied), how can we efficiently direct the nearest unoccupied unit to respond upon receiving an incoming call? As taxis are continually moving, how can we keep the information in the database current in an efficient manner? Secondly, if updates occur only intermittently, what is the most accurate approach to answering queries when data may be stale? Third, can we identify any interesting movement patterns? Can we identify points of interest from common stops made throughout the city? Is it possible to infer the efficient routes for a particular origin and destination given historical movement patterns?

The examples presented above introduce problems in three different areas of managing spatiotemporal data: (i) querying and indexing, (ii) tracking, and (iii) mining. Querying mobile objects, like any temporal data, introduces challenges in defining expressive predicates that properly handle the time domain. Additionally, constructing and maintaining an index structure to efficiently process queries over mobile objects is difficult due to the high frequency of necessary updates. Because the values (i.e. location) are constantly changing, the query workload is skewed to become update-centric, forcing the index to update its struc-

ture more frequently. Unless the index is specially designed for such a query workload, frequent updates can be very costly and even outweigh any benefit the index structure provides for query processing.

Tracking is critical to managing both the spatial and temporal uncertainty in an object's position. Accurate tracking is challenging, especially at the database scale (i.e. tracking thousands to hundreds of thousands of objects), due to the computational constraints. Inferences and predictions about an object's position must be made quickly, and should use all of the data that has been observed thus far.

The difficulties in mining for patterns in spatiotemporal data are similar to those mentioned for querying. Core mining problems, such as that of identifying groups or clusters, is made significantly more difficult when the data change positions over time. New definitions and objectives must be defined which take into account not only the current data configuration, but also the past (or predicted) configurations.

Additionally, the problem of data uncertainty is inherent in all areas of managing spatiotemporal data. Despite technological improvements, the ability to localize mobile objects is still only available up to a degree of error. Due to the nature of dealing with inexact data, new approaches to indexing, querying, and mining are necessary to effectively account for ambiguities in the data [32, 15, 82, 31]. Although data uncertainty spawns from a variety of sources, it can be broadly categorized as one of two types: *spatial* and *temporal* uncertainty.

- **Spatial Uncertainty** is uncertainty in the location of an object at the instance an observation is made. That is, spatial uncertainty describes the limitations of a sensor to provide an accurate reading of an object's position. For example, high quality GPS sensors typically provide measurement accuracy in the range of $1 - 10$ meters, lower quality hardware is in the range of $10-50$ meters, and localization from cellular tower triangulation may resolve position to only within $100 - 2,000$ meters [6, 79].

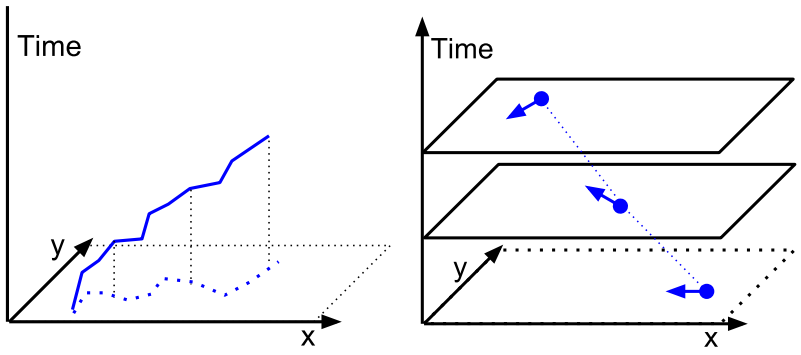
- **Temporal Uncertainty** is the uncertainty in an object's position since the previously received update. Temporal uncertainty arises due to the update schedule of how frequently an object will send information about its position to a database. Since objects may move continuously but only report their positions intermittently, there is a time-lag in which the database contains stale information. In several datasets, GPS traces have shown incredible variance in the frequency with which measurements are provided. The temporal resolution ranges from very high (1 second intervals) to very low ($>$ 2 min. intervals) [97].

In this chapter, we introduce the types of problems that arise from managing spatiotemporal data and survey the recent research that addresses these issues. In our review, we attempt to cover work on data management, object tracking (processing updates of moving objects), and mining spatiotemporal data. Our aim in this chapter is twofold: (i) to provide a review of the recent developments in each of the different areas of study relating to spatiotemporal data and (ii) to introduce work on tracking and show how it relates to the database-centric research (e.g. querying, indexing, and mining). While we attempt to provide a broad overview of recent work in all of the mentioned areas, we pay special attention to work which explicitly manages uncertainty in the spatiotemporal data.

The rest of the chapter is organized as follows: first we will review work on data management for spatiotemporal data, including indexing and querying, in section 2. Next, we introduce the problem of tracking and review some core and recent developments in that area in section 3. In section 4, we review some recent work on mining spatiotemporal data in three broad categories: (i) clustering, (ii) popular route discovery, and (iii) identifying mobility patterns. We conclude by discussing directions for future research.

2. Data Management for Mobile Objects

Database management systems for spatiotemporal data can be characterized as one of two types: *spatiotemporal database systems* (STDB) and *moving object database systems* (MOD). Both are used to manage data collected from mobile objects, however, the specific problems each solves is quite different. STDBs store the complete historic trajectory of each object, and thus allows users to answer complex queries about user movement *over time*. For example, "find all users that passed through region A between 8 - 10am, through region B between 1 - 2pm and region C between 4 - 7pm". In contrast, MODs only maintain the current position of each object along with each object's velocity or heading information if it is available. Therefore, while STDBs contain more complete information, MODs provide access to location data that is consistently current (see figure 2). MODs are well suited to answer queries about the current (and near-future) configuration of mobile objects. Both types of database systems present a unique set of challenges due to the data they manage. In this section, we introduce both systems, as well as some recent work which addresses the challenges in efficient indexing and query processing. In addition to the general issues with managing spatiotem-



(a) Trajectory Object (b) Moving Object

Figure 10.2. Different types of data that may be extracted from mobile objects. In (a), a full trajectory is stored which describes an object's movement in space over a historic time interval. In (b), the position and velocity of an object are stored at the current time, however, historic information is not maintained.

poral data, we also introduce some specific problems in handling data uncertainty.

Many of the index structures for spatiotemporal trajectories are based on the R-tree [28, 5, 73]. For a comprehensive review of spatio-temporal indexing methods we refer the interested reader to [57] and [55].

2.1 Spatiotemporal Database Systems

A STDB allows the user to efficiently query the historic movements of a set of objects over a period of time (in the past). Although queries over spatiotemporal data can be quite complex [29], the basic queries that every STDB should answer are time-interval *range* and *nearest-neighbor* queries. A time-interval range queries answers the question "which objects traveled through region R between the times t_{start} and t_{end}?" (note that t_{end} is some time point in the past). For instance, this could be used to find out which vehicles traveled within a city during rush hour. The semantics for the time-interval nearest neighbor query may change slightly between systems, but the basic idea is to return the set of objects closest to some query point over a given time interval.

Pfoser et al. [66] distinguish between two types of queries on trajectories as **topological queries** and **navigational queries**. Topological queries are concerned with finding a set of trajectories that satisfy some spatial and temporal constraint. Range and nearest-neighbor queries over a time slice or interval are prime examples of topological queries. Navigational queries are based on derived information extracted from the trajectory and may involve dynamic information about the objects

such as *speed* and *heading*. For example, it may be interesting to identify an object's top speed over a time interval, or find the set of objects with a particular heading at a specified time. Combining topological and navigational queries provide a powerful approach for analyzing complex spatiotemporal patterns.

To efficiently answer topological queries over trajectories, Pfoser et al. [66] introduce two index structures, the spatiotemporal R-tree (STR-tree) and the trajectory bundle tree (TB-tree). both structures naturally extend the basic R-tree [28] to handle trajectories in a more efficient manner. The problem with directly applying the R-tree to index trajectories is the amount of *dead space* typically incurred. To maintain a nearly $O(logN)$ access time, whole trajectories must be stored as single units; this reduces the discriminative power of the index structure due to the large area necessary to bound such a region. Alternatively, splitting each trajectory into a set of line segments improves the discrimination of each bounding box, however, access time is now dependent on the length of each trajectory (i.e. access time $O(klogN)$).

The approach utilized in the STR-tree is to segment trajectories while keeping parts of the same trajectory close within the index structure to improve the efficiency of queries over large time intervals. As compared to the R-tree, the major contributions of this index are new *insertion* and *split* methods which provide different optimization criteria. The authors assume that each trajectory is associated with a unique identifier and has already been partitioned so the index may already contain a partial trajectory for an object. The insertion algorithm first attempts to fit the new segment in the same minimum bounding box (MBR) as the previous segment from the same trajectory. We may split this index node if it is full as long as the parent is not full. Otherwise, the trajectory segment is inserted using the original *ChooseLeaf* algorithm, which locates the leaf node for which inserting the current segment will incur the lowest cost in terms of MBR overlap, area and dead space [28]. The split algorithm also considers how pairs of trajectory segments are related when optimizing the split. Segments that are not related to any other segments in the node (i.e. they come from different objects), may be placed into the new node, otherwise, if all segments are related the node is partitioned by time such that the newest segments will remain together. In general, the insert and split procedures first optimize for trajectory preservation (i.e. keeping segments of the same object nearby in the index structure) and then for spatial closeness (i.e. minimizing the increase in an MBR).

In contrast to the STR-tree, the TB-tree takes a more dramatic approach and groups segments of the trajectory to *ensure* that they are all 'bundled' together for fast retrieval. The leaf nodes of a TB-tree are

forced to only contain segments belonging to the same trajectory, thus making much larger concessions in MBR overlap. The insertion strategy for the TB-tree is simply to find the previous segment of the trajectory being inserted and place this segment in the same MBR. If this is not possible, instead of splitting which would break apart segments from the same trajectory, a new node is constructed for the segment and is inserted in the first non-full parent. All of the leaf nodes containing a trajectory are connected through a doubly-linked list so that an individual object can be retrieved from any individual segment.

As compared to a 3-dimensional R-tree, both the STR-tree and the TB-tree result in more compact index structures with space utilization, or how tightly packed the nodes of the index structure are, approaching 100%. Additionally, both structure provide improved query times for range queries over a small numbers of moving objects, however, as the number of objects increased, the R-tree typically provided fewer node accesses. However, for *combined queries*, which retrieve trajectories through various means, the TB-tree outperformed the STR-tree and the R-tree by an order of magnitude even for a large number of mobile objects.

Chakka et al. [11] introduce a scheme for indexing spatiotemporal trajectories called *Scalable and Efficient Trajectory Index* (SETI). The idea proposed by the authors is that splitting the space dimensions from time, allows spatially close trajectory segments to be grouped together *and then* indexed by time, which provides a more compact and efficiently searchable structure. Their approach is to first statically partition the space over which objects move and then maintain an index structure only over the time intervals of segments contained within each disjoint spatial partition. Each trajectory segment in a space partition is indexed with an R-tree by a two dimensional point representing the start and end time of that segment.

Queries are processed using a spatial and temporal filtering followed by a refinement step to construct the answer set. The spatial filtering simply returns all of the spatial partitions contained within the spatial partitions that are contained within (or overlap with) the query region. The subsequent temporal filtering simply poses a range query to the R-tree in each spatial partition to retrieve the individual trajectory segments. A refinement step is necessary only over those spatial cells that partially intersect with the query region.

This conceptually simple idea of splitting the time and space dimensions was shown to provide significant improvements in query processing time over TB-tree. Experimental results show that SETI consistently provides lower query processing times for spatial range queries over time

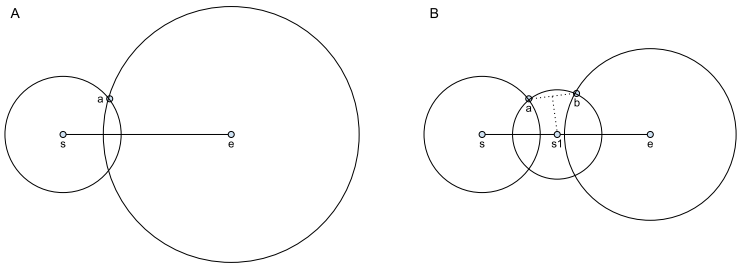


Figure 10.3. In (A), only point a has been processed and is thus the NN for both SPs s and e. In (B), after processing point b, we update the NN of e and create a new SP $s1$.

intervals (as well as time slices). However, it is not clear how SETI compares to TB-tree for *combined queries*, in which the query is both topological and navigational (mentioned in [66]), where processing may be required to access different segments of the same trajectory (which TB-tree does efficiently). A similar approach was simultaneously developed by Song et al. [75] in which the authors split the space and time dimensions.

In addition to the range query, nearest-neighbor and k-NN queries are also fundamental for any spatiotemporal data management system. For identifying nearby objects in the context of trajectory data, the continuous nearest-neighbor (CNN) query has been proposed in which a sequence of nearest neighbors are returned such that the nearest-neighbor (NN) is known for every time interval [27, 77]. Tao et al. [77] introduce the CNN query and develop an efficient query processing algorithm. The objective in processing a CNN is to identify, for every time range within the query interval, the nearest object to the query trajectory. The authors use a geometric approach in which a set of *split points*, locations at which the NN of the query trajectory changes, is maintained and incrementally updated during processing. Trajectories are processed by considering each line segment separately and aggregating the results through post-processing. The algorithm starts with the end points of the line segment being the two split points (SPs). Objects (i.e. spatial points) in the database are processed sequentially. For each object, we first check if it is the NN to any split points. This is done by maintaining a circle centered at each split point such that the radius is the distance to the closest (known) object. If a new object lies within this circle, then we must update the SPs and adjust the radii. Updates to SPs are made by computing the perpendicular bisector, the point at which this line intersects the trajectory will become a new SP. This process is shown in figure 2.1.

More recently, Guting et al. [27] introduce algorithms that provide more efficient query processing efficiency of CNN queries. The authors assume the trajectories are indexed using a $3d$ R-tree and they develop a *filter-and-refine* approach for processing queries which uses geometric pruning techniques. In the filtering step, we traverse the R-tree, pruning nodes from the candidate set based on the geometrical properties of the bounding box. The refinement step performs a modified plane-sweep algorithm over the candidate set of trajectories to test over what time intervals, if any, the trajectory is the nearest-neighbor to the query. The authors experimentally show that their method offers significantly faster query processing time and fewer disk accesses as both the size of the database and the query range increase as compared to other CNN processing algorithms.

Trajcevski et al. [81] also develop a query processing algorithm and index structure for the CNN query, however, they address this query when trajectories are uncertain. An uncertain trajectory is modeled as a cylinder in which the mobile object may have traveled anywhere within the enclosed area (i.e. the object's true location is assumed to be uniformly distributed within the cylinder). Further details about the uncertainty model for trajectories and how it is used in general query processing is provided in [80, 82]. The authors first extend the earlier work by allowing queries over pairs of objects in the database instead of known regions of interest by specifying a query as an uncertain trajectory [82]. The authors then show that they can order objects by their probability of being the nearest neighbor to the query (at any given time) by ranking objects according to their expected distances to the query location for any symmetric distribution function. This provides an ordering in which to process the objects (using a similar methodology as introduced in [15]).

To identify which objects are the nearest neighbors over a given time interval, the authors of [81] propose a hierarchical data structure such that for each node in the tree, that object has the highest probability, besides the parent, of being the nearest neighbor to the query within the time interval bounded by the parent. That is, the first level of the tree would be the nearest neighbors, each over its disjunct time interval. The second level would be nodes that are second nearest neighbors over disjunct time intervals bounded by their parent, and so on. They show how to construct this structure and use it to answer the NN query and several deviations.

In addition to range and nearest-neighbor queries, other, more complicated, queries have been defined on spatiotemporal data [4, 29]. For instance, Bakalov et al. [4] introduce a spatiotemporal join query that

finds all pairs of objects with (nearly) matching subtrajectories over a given time interval. This query could be useful in clustering subtrajectories or simply identifying groups of mobile objects that traversed similar paths. More formally, the authors define the spatiotemporal join query to take two sets of trajectories, a distance threshold, ϵ, and time interval, δ_t, and returns all pairs such that there exists a subregion of each trajectory of length δ_t where the distance between the subregions is at most ϵ. The query is called a time relaxed spatiotemporal trajectory join (TRSTJ), because the query only constrains the length of the trajectory subregion, not the specific start time. To answer the TRSTJ, the authors propose a filter and refine approach where they use a compact trajectory representation and lower bound the Euclidean distance between trajectories.

2.2 Moving Object Databases

Unlike STDBs, a *moving object database* (MOD) only stores the *current* position of each object. MODs constantly contain up-to-date information about the location of each object and are therefore useful in real-time applications of managing a large number of mobile objects (e.g. navigation or emergency response dispatch). Similar types of queries are supported on MODs, however, the time intervals over which the data may be queries is limited to the present and future. Instead of asking where an object *has* been, a MOD answers the query "where is object A now"? or "where will it be in 5 minutes"? In order to answer such queries, the system must have some method to predict the future location of each object given its current location and its velocity. We will cover different approaches for updating and predicting location information in detail in section 3.

Similar to STDBs, a primary difficulty for MODs is constructing and maintaining an index structure. However, the *cause* of the difficulty in the two systems is quite different. Here, the problem is keeping the location information for all objects up to date, which requires frequent updates to the index. Continually modifying an index structure is likely to cause the discriminative capabilities of the structure to degrade over time unless special care is taken.

Cheng et al. [15] introduce a method for answering range queries and nearest neighbor queries with probabilistic guarantees when the location of objects is uncertain. The authors propose an uncertainty model that specifies a bounded region within which a mobile objects may be located with equal probability. The authors were the first to define and propose a solution to the probabilistic nearest neighbor query (PNNQ). They

use a filter-and-refine approach in which objects located far away can be pruned (i.e. their shortest distance is larger than the longest distance of a closer object). After filtering objects that obviously do not satisfy the query, the space in which objects need to be evaluated (i.e. over which the integral needs to be performed) can be bounded. Lastly, the remaining objects are evaluated by computing the integral in Eq. 10.1.

$$p_i = \int_{n_i}^{f} p_i(r) \prod_{k \neq i}^{|S|} (1 - P_k(r)) dr \tag{10.1}$$

In Eq. 10.1, n_i is the shortest distance from object O_i to the query point q, and f is the bounded region over which objects must be evaluated. The integration can be interpreted as the probability that O_i is at a distance of r from q while all other points in the candidate set, S, are at a distance greater than r, which is evaluated over all possible values of r. The authors introduce a more efficient approach to evaluating this integral by sorting S and breaking up the integration limits to only the region in which $p_i(r)$ is non-zero. Furthermore, they also utilize an R-tree based index structure, called the velocity constrained index (VCI) [68], to improve processing time for large datasets.

Saltenis et al. [71] introduce a general index structure for efficiently processing queries on mobile objects called the time parameterized R-tree (TPR-tree). The idea behind the TPR-tree is to allow MBRs to grow and change position as a function of time in order to reduce the number of necessary updates to the index structure from objects moving (see figure 2.2). The authors propose *conservative bounds*, in which the MBR expands by the maximum velocity of the contained set of objects in each dimension. That is, considering the number line in which values to the left are smaller and those to the right are larger, the MBR expansions in both directions of one dimension are given in Eq. 10.1.

$$\begin{aligned} \overleftarrow{x} &= \min(o_i.\text{pos}(t)) - \min(o_i.\text{vel})(t_{\text{diff}}) \\ \overrightarrow{x} &= \max(o_i.\text{pos}(t)) + \max(o_i.\text{vel})(t_{\text{diff}}) \end{aligned} \tag{10.2}$$

Non-leaf level MBRs are constructed by aggregating the bounds from each of their children. For each dimension, the expansion in each direction is the maximum over all of its children.

Although the TPR-tree may be used to index mobile objects indefinitely, it is optimized only for a particular time horizon, after which the performance of the index may deteriorate. Specifically, the index structure requires two parameters: the querying window, W, which defines how far queries may look into the future and the index usage time, U, which defines how long users will query the index. Combining both of

these values we get the time horizon, $H = U + W$, which is the time over which the index structure must be able to answer queries. Using these values, the index structure can be optimized since there is a limit as to how far into the future the index will be required to answer queries.

The TPR-tree uses the same optimization concepts from the R*-tree [5], specifically the area of MBRs, perimeter length, and overlap area, except they are minimized over a time horizon instead of simply minimizing the current state of the index. If we consider an objective function that minimizes overlap between MBRs $A(t)$, then we would like to optimize this function over the given time horizon: $\int_t^{t+H} A(t)dt$. The operations used in the TPR-tree are all analogous to the R*-tree, with the only difference being the function *splitNode*(), which determines how objects contained in full nodes should be partitioned. Since MBRs are dynamic, the TPR-tree split is based on optimizing over both the current positions *and* velocities.

Query processing in the TPR-tree is similar to the R-tree [28]. Since all of the MBRs are parameterized by time, answering a time-slice range query remains completely unchanged. To answer a time-interval range query or a moving range query, it is necessary to identify intersections between the MBR and the query region over the specified time interval in each dimension and check that the intersections happened over the same time interval in all dimensions. This is computed in a straight forward manner by comparing line segments of the query region and MBR boundaries in each dimension over time (the lines describing movement in the both the x and y dimensions over a time interval).

The simplicity of the TPR-tree has made it a popular choice for indexing MODs. Tao et al. [78] have extended the original work by introducing the TPR*-tree, where they improve upon the optimizations introduced in [71]. The TPR*-tree tightens the bounds of the MBRs thus making the index more discriminative. Experiments show substantial improvements in performance over the TPR-tree for large datasets. To handle spatial uncertainty in mobile objects, Hosbond et al. [31] adapt the TPR-tree to index moving objects with inexact location information. The authors essentially model location uncertainty as a Δ term in which each object can vary some amount from its stated position. They incorporate this error term into the MBR parameterization to guarantee each object is properly bounded over time.

In [98], the authors present both a movement model for mobile objects with uncertainty as well as an index structure for efficient query processing. The proposed uncertain moving object model defines a probability distribution over location and velocity (independently). The model works by griding the space at some resolution and assigning each cell

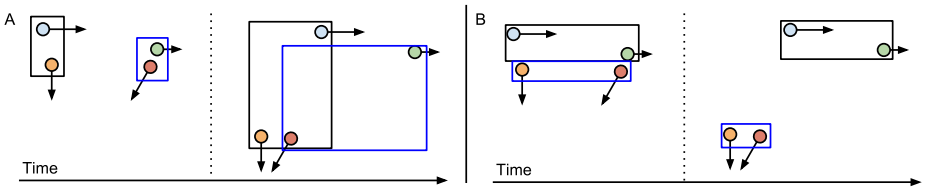


Figure 10.4. An example of how the TPR optimizes MBRs over time. The right panel (A) shows the MBR optimized for the current configuration, whereas the left (B) considers both the current object position as well as velocity.

a probability thereby constructing a discrete distribution over possible locations (velocities). Combining the current location and velocity distributions, it is then possible to predict an object's future location. Utilizing a gridded space technique allows the representation of arbitrary distributions, however, it also incurs a heavier cost for prediction since sampling must be employed. The authors then adapt the B^x-tree [35] to handle objects with uncertain locations. For each object, every non-zero probability grid cell of the object's location is inserted into the index structure, thus accounting for location uncertainty. Thus a trade-off is made between location accuracy and the space consumption and search time of the index structure.

Chung et al. [17] develop query processing and indexing techniques to efficiently answer probabilistic range queries when uncertainty in the trajectories is described with Gaussian noise. The authors assume that objects are moving along a known path and thus treat each object as moving in 1-dimensional space over time. The movement uncertainty for each object is modeled by Brownian motion in which the object's velocity is normally distributed. To avoid the linear scan of the database, the authors apply the Hough transform, which maps lines to points, to the expected trajectory of an object and use an R-tree to index these points in the dual space. Queries are then transformed in the same manner, although some additional work is required since the trajectories contain some location uncertainty. The query region is expanded depending on the probability threshold issued in the query. Although this approach provides efficient processing of probabilistic range queries, it makes the limiting assumption that objects move in 1-dimensional space.

Chen et al. [12] perform an experimental investigation of the effectiveness of several index structures for MODs under various conditions. The authors compare the TPR-tree [71], TPR*-tree [78], B^x-tree [35], B^{dual}-tree [95], STRIPES [65], and RUM*-tree [92]. The R-tree with Update Memo (RUM*-tree) employs main-memory memos to help avoid

frequent disk accesses for deleting old record entries. This reduces the cost of an updating an object's position simply to that of inserting a new object into the R-tree and old entries are deleted in batches. The B^{dual}-tree and STRIPES are both indexes that utilize dual space. The B^{dual}-tree maps an objects location and velocity to a 1-dimensional point using a Hilbert curve. STRIPES maps a mobile object to a four dimensional point and applies a quad-tree based structure to index the space. The experimental evaluation provides a thorough comparison between the different index structures on several aspects of updating and querying mobile objects. Although each index performs well under certain circumstances, the B^x-tree consistently performed near the top over all experiments never costing more that 2X the best index structure. While the TPR-trees were *the* most efficient indexes for the querying experiments (in terms of time and I/O), they also performed the worst for updates. Overall, this work highlights the need to clearly identify the expected query workload distribution in order to select the most efficient structure.

A problem that has received far less attention in the spatiotemporal data management community is that of how to most efficiently update the database with new positions for mobile objects. Wolfson et al. [90] address this problem by framing it as an optimization problem. The authors provide a model of the cost to poll an object in order to update its location and a cost for answering a query given the current (estimated) location uncertainty (i.e. how much error are we willing to tolerate) and aim to minimize their total costs while handling a query workload. The authors also introduce a spatial indexing scheme for moving objects that considers the possible location error. For this, they use an R^+-tree [73] over three dimensions (the x, y plane and time). The index is updated only when objects report their position to the database (using the derived update policy).

2.3 Mobile Objects on Road Networks

The queries and index structures discussed thus far consider the case of unrestricted movement, however, movement is often restricted to a transportation network (i.e. road network). If the structure of the underlying transportation network is known, then incorporating this additional information can provide more accurate tracking and prediction of object locations (and therefore more meaningful queries). Restricting movement to the network structure brings about its own set of challenges.

Zheng et al. [100] address the problem of managing moving objects on road networks where the specific path traveled by an object is uncertain. The authors represent the uncertain locations of objects as time-dependent probability distributions and develop methods to efficiently query the objects. They represent a road network as a graph in which the edges have two attributes, the length of the road and the maximum allowed speed along that path. There are two kinds of uncertainty in this work: *path uncertainty*, that is, which path (sequence of edges) did the object take to get from the last position to the next, and *location uncertainty*, given the path, the actual location of the object at time t. Path uncertainty arises from the object selecting a specific route due to some criteria (i.e. traffic, road work, landmarks) and location uncertainty arises from the fact that we don't know how fast the object was moving the entire time, though we can bound this by the maximum allowable speed on each segment.

In [100], the authors propose a novel index structure for processing range queries on uncertain objects over road networks called the Uncertain Trajectory Hierarchy (UTH). The UTH consists of three levels: (i) a network edge hash table, (ii) a movement R-tree, and (iii) a trajectory list. The edge hash table allows fast retrieval of any specific road segment. The movement R-tree is an index over time for a specific road segment which allows us to identify the set of objects traveling along a road over a given time interval. Lastly, the trajectory list stores the actual trajectory for each object. A filter and refinement approach to answering uncertain path queries on road networks is developed by the authors by leveraging the UTH index.

Hua and Pei [32] also study the problem of answering path queries on road networks, however, in this case it is the edge speeds (or flow) that are uncertain, not the positions of the moving objects. The authors introduced a network model in which adjacent edges in the road network were correlated and they introduce exact and more efficient approximate methods for computing the probability of paths. They also provide an A^*-like algorithm for efficiently finding the most probable paths under specific speed constraints.

Kim et al. [43] develop an index structure for objects with movements that are restricted to a road network called *Indexing Moving Objects on road sectors* (IMORS). The index is composed of a static component, that is made up of an R^*-tree over the road segments, and a dynamic component, which contains a mapping from road segments to mobile objects. The dynamic module of the index structure is essentially a table indexed by object identifiers' that contains attributes of the object as well as its location and a pointer to the road segment it currently

occupies. To process an update, the object record is found, its location is updated and we check if it has changed road segments. If so, we search the R-tree based on the new coordinates and fix the pointer in the object record to point to the updated road segment. The IMORS provides significant efficiency gains in processing updates over the TPR-tree (to which it was compared in the experiments), by utilizing the static nature of the road network. Since locations of roads do not move, object positions are modified by updated the road segment to which they point, however, all searching is done over the road network indexed R^*-tree.

When dealing with mobile objects that have restricted movement, such as automobiles on a road network, this gain in efficiency in indexing and query, and increased accuracy in predicting future locations is common. The idea of integrating all available information about the object movement is important as it allows us to identify object positions and movement more reliably. Utilizing extra information can help combat against the problems of spatial and temporal uncertainty when managing spatiotemporal data.

Additionally, there is a plethora of work in the area of efficiently processing routing queries for navigation queries. For instance, computing the quickest (or most efficient) route between two points considering the dynamics of the road network is of great interest. Incorporating extra information like speed limits, predicted traffic patterns, and road closures can greatly improve routing and therefore help relieve traffic congestion.

Nikolova et al. [63] show several theoretical results related to problems of optimal route planning. The authors show the surprising result that the problem of finding an optimal route *and* start time can be solved using standard shortest path algorithms for certain cost functions while the optimal route planning problem when the start time is fixed is NP-hard. Similarly, Wilkie et al. [89] consider the routing problem on a stochastic traffic network, however, they integrate the effect of the planner into future predictions. That is, vehicles will query the planning system to ask for directions, assuming the vehicles stick to these routes, the planner has additional information about the state of the traffic in the future. Malviya et al. [54] introduce an approach for continuously monitoring a set of shortest path for mobile objects. The authors first precompute a set of *good* routes using a road network and historic travel times and then re-rank these paths by integrating information about real-time traffic. Gonzalez et al. [24] developed a fastest-path algorithm which utilizes historic information about traffic and weather conditions to provide reliable routes. The authors also introduce a network partitioning algorithm which reduces the search space by focusing on the

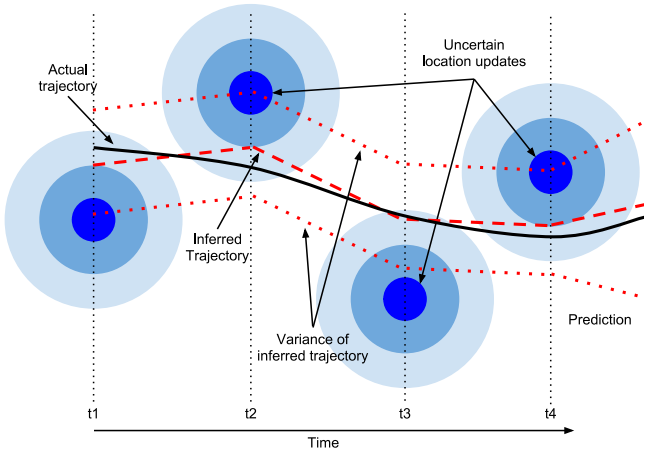


Figure 10.5. An example of turning noisy location observations into a trajectory. The filtering model attempts to identify the most likely path that would have generated the noisy observations given a specific movement model (linear in this case: $X_t = X_{t-1} + \dot{X}_{t-1}\ (\Delta t)$).

most traveled roadways (i.e. highways) and only expanding extra edges when they have historically exhibited improved performance.

3. Probabilistic Models for Tracking

Processing location updates from mobile objects is a crucial component of managing spatiotemporal data because the raw locations obtained from a sensor are often noisy. Even GPS has been shown to contain errors on the order tens or hundreds of meters [6]. Because locations at adjacent time steps are not independent (see figure 3), it is possible to incorporate information about the dynamics of the mobile object in order to improve upon its current position. This is exactly what *tracking* accomplishes. By explicitly modeling the dependencies between locations observed at adjacent time steps, we can filter the raw data to produce a cleaner estimate of the object's trajectory which accounts for noise in the sensor as well as the system dynamics (e.g. friction). Additional constraints over an object's possible movements (e.g. road network) can also be incorporated to further improve the filtered trajectory.

Due to the inherent uncertainty in the problem of tracking mobile objects, probabilistic models such as dynamic Bayesian networks (DBNs) have been applied to several tracking scenarios with great success [50, 60]. In this section, we will first introduce the basic problems involved in tracking mobile objects. We pose the tracking problem as Bayesian

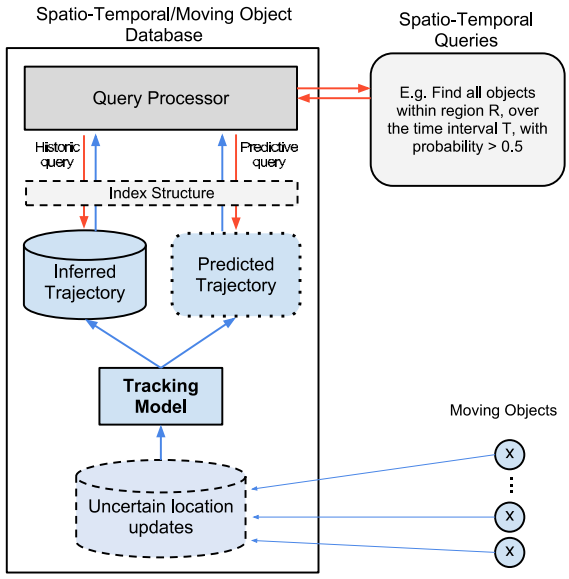


Figure 10.6. The (general) architecture of a STDB or a MOD. Both systems require an algorithm to turn the raw location estimates from the moving objects into a usable trajectory. In the case of the MOD, this conversion happens in real-time as new locations are made available (filtering) and in the the case of STDB, the complete sequence of locations and time stamps are available (smoothing). The quality of the inferred locations directly affects query accuracy, thus the tracking algorithm is a vital component of the data management system.

filtering task, then we review the Kalman filter model (KFM) [39, 88] in detail. Lastly, we discuss some methods which address the tracking problem when objects are constrained to move on a road network. Table 10.1 contains a list of notation used throughout this section.

3.1 The Tracking Problem

The general task of tracking can be formulated as a Bayesian filtering problem where we would like to estimate the value of an unobserved random variable x, given an observation z. Because x defines the state of a mobile object (position, velocity, altitude, etc.), this value will change over time and we would like to re-estimate it each time a new observation becomes available. The general problem of Bayesian filtering is then to update our beliefs about x_t, incorporating all available information (i.e. $z_{1:t}$) by computing the posterior distribution $p(x_t|z_{1:t})$. To keep our presentation clear, we will assume that the state of a mobile object, x, is described by a vector containing the object's current position and ve-

Table 10.1. Notation

Notation	Explanation
x_t	Belief state (hidden) for at time t
X	Location component of belief state
$\dot{X}$	Velocity component of belief state
$z_{0:t}$	Observations from time 0..t
A	Deterministic transition matrix
H	Deterministic observation mask
Q	Covariance matrix for the movement dynamics. Determines the amount of uncertainty we have in our transition model.
R	Observation covariance matrix. Determines the amount of uncertainty we have in our observations.
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with expected value μ and covariance matrix Σ.

locity ($x_t = [X, Y, \dot{X}, \dot{Y}]$) and the location measurement, z_t is described by a position ($z_t = [X, Y]$).

$$\begin{aligned} p(x_t|z_{1:t}) &= \frac{p(z_t|x_t)p(x_{t-1}|z_{1:t-1})}{p(z_{1:t})} \\ &\propto p(z_t|x_t)p(x_{t-1}|z_{1:t-1}) \end{aligned} \tag{10.3}$$

The first term in Eq. 10.2 is the likelihood function which describes the relationship between x_t and z_t (i.e. describes the sensor error). For example, GPS sensors are typically assumed to have error that is normally distributed about the true location. In this case, $p(z_t|x_t) = \mathcal{N}(z_t; x_t, \sigma)$, where σ describes how much variation we expect to see in the measurement.

The second term, $p(x_{t-1}|z_{1:t-1})$, is the posterior distribution of x_{t-1}. That is, this term is the result of filtering at the previous time step. From this equation, we see that it is possible to recursively update our belief about the state of a mobile object online (as new data arrive). All of the information about x_{t-1} is captured in $p(x_{t-1}|z_{1:t-1})$, thus there is no need to revisit old datum. Lastly, the denominator is the marginal probability of the sequence of observations. Since the observations remain fixed (this data is observed), this term may be considered a normalizing constant and ignored for our purposes. For a readable introduction to Bayesian statistics in a more general context, we refer the interested reader to [30].

While tracking is an *online* problem (i.e. updates must be made as data arrive), in general there are three types of inference we may be interested in for any DBN: (i) prediction, (ii) filtering and (iii) smoothing.

A graphical representation of each type of inference for a simple chain model with a single hidden variable is shown in figure 3.1. Prediction and filtering are computable online, while smoothing requires observations from future instances in order to *correct* our estimate of an object's state given more information. In the context of data management systems, filtering and prediction would be used in a MOD, while smoothing would be used to obtain complete historic trajectories of mobile objects for a STDB.

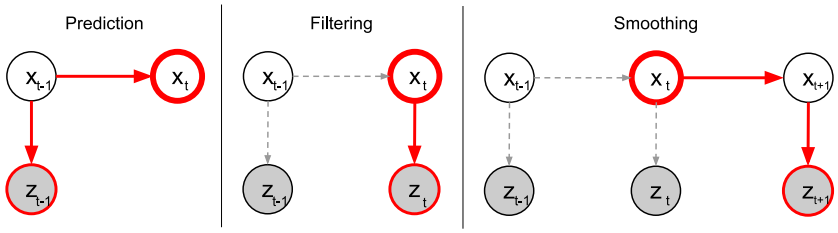


Figure 10.7. KFM Inference steps: **Prediction** predicts the value of x_t from the last known position of the object, z_{t-1} and the movement model. This is computed in the graphical model by integrating out the unobserved value of x_{t-1}. **Filtering** *corrects* the value of x_t by incorporating the latest uncertain observation, z_t. *Smoothing* updates the filtered estimate for x_t by also incorporating information from later observations ($z_{1..T}$).

In general, it is rarely possible to evaluate Eq. 10.2 exactly, and approximate methods have become quite common [13, 19, 59]. However, under certain modeling assumptions, exact inference is tractable. Next, we introduce the Kalman filter model (KFM), a popular model for which exact inference is tractable. Due to the popularity of the KFM and its widespread adoption in tracking and sequential data processing [40, 41, 74, 91, 51, 94], we discuss this model in some depth. Our objective in the following section is to briefly introduce the Kalman filter to unfamiliar readers, including some intuition as to how and why the model works.

3.2 Kalman Filter

The Kalman filter model [39] (KFM) is a linear dynamic system that offers an efficient exact inference procedure. The efficiency stems from the fact that all of the variables in the model are assumed to come from a joint Gaussian distribution. As a result, both the observation noise (Eq. 10.4) and the system dynamics (Eq. 10.5) are assumed to be Gaussian distributions. The observation noise describes how observations are related to the actual belief state. In this case, we expect observations to be distributed normally about the true state with the degree of

uncertainty given by the covariance matrix, R. The system dynamics describe how the state changes between time steps. In the KFM, we assume the current state is a linear function of the previous state (e.g. $x_t = x_{t-1} + \dot{x}_t$).

$$p(z_t|x_t) = \mathcal{N}(x_t, R) \tag{10.4}$$
$$p(x_t|x_{t-1}) = \mathcal{N}(Ax_t, Q) \tag{10.5}$$

Eq. 10.5 describes the model dynamics and Eq. 10.4 describes the noisy observation process. The transition probability of the previous belief state x_{t-1} to the current belief state x_t depends on the deterministic transition matrix A and the transition covariance matrix Q. To explain the ideas behind the Kalman filter we consider a simple example in which we are tracking a moving object in one-dimension. The representation of our belief state, x, contains the object's current location and velocity. Therefore, the transition matrix $\mathbf{A}$, is defined to encode $X_t = X_{t-1} + \dot{X_{t-1}}t$ and $\dot{X}_t = \dot{X_{t-1}}$, that is $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. The covariance matrix Q describes the noise in the dynamic process, such as head/tail winds, loss of power due to friction, changing altitudes, etc. Suppose that we are only able to observe the object's location at discrete time steps, but have no way of acquiring the velocity directly. In this case the deterministic observation mask $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$, and we must infer velocity completely from the location observations. The covariance matrix R, describes how much uncertainty exists in these observations. For example, if we were using a GPS sensor to track the object, R would be relatively small, allowing the observations to vary only a couple of meters from the actual location. In contrast, if we relied on cellular tower triangulation to track the moving object, R would be very large, allowing observed locations to be hundreds of meters from the true location.

Since the state variable, x_t is assumed to be normally distributed, we need only maintain the mean and variance of the distribution at each time step to completely describe our knowledge about x_t. Because of this, and some nice analytic properties of the normal distribution, simple update equations for the necessary parameters are tractable. This simplicity makes the KFM a popular choice for modeling dynamic, real-valued data. Because of its popularity, in this section we will describe the KFM filtering and smoothing algorithms and provide the reader with some intuition as to how the update equations work. Before getting started on the KFM, we first digress slightly to review some important properties of the Gaussian distribution.

3.2.1 Joint, Marginal, and Conditional Gaussians. In this section we interchange the terms normal and Gaussian distribution. In both cases, we are referring to the same density function (shown in Eq. 10.6).

$$p(x_1, ..., x_\rho) = \frac{1}{(2\pi)^{\rho/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right) \quad (10.6)$$

For any set of random variables that are jointly normally distributed, all marginal and conditional distributions associated with any individual or subgroup of variables are also normally distributed. For example, consider the simple bivariate normal distribution $p(x, y)$ with parameters given in Eq. 10.7.

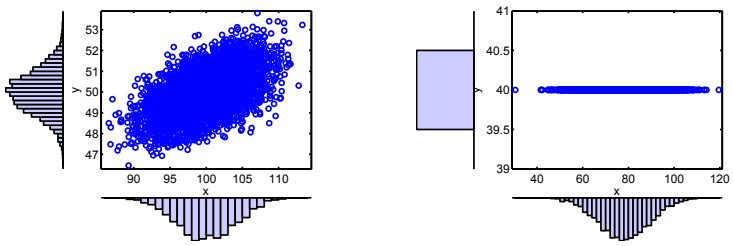
$$p(x,y) = \mathcal{N}(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{x,y} \\ \Sigma_{y,x} & \Sigma_y \end{bmatrix})$$

The marginal distribution for x is computed by integrating over all possible values of y. That is, $p(x) = \int p(x, y)\ dy$. For a joint Gaussian, marginalization or *integrating out* variables is a simple procedure, since the result is a normal distribution we must only find the mean and covariance matrix that specify the distribution. In this case, we simply take the mean and covariance sub-matrix corresponding only to the variable(s) of interest and the marginal is again normally distributed. For example, using the joint in Eq. 10.7, $p(x) = \mathcal{N}(\mu_x, \Sigma_x)$.

Marginalization is the process of simply removing a variable from our distribution. However, this process does not provide us with any additional information about the variable of interest, it only serves to simplify our distribution form by reducing dimensionality. Alternatively, it may be possible to observe the value of a variable may provide us with information about our variable of interest. In this case we are interested in computing the *conditional distribution*. That is, the distribution over x given that you have observed a specific value for y. In the case of jointly distributed Gaussian variables, the conditional distribution is again a Gaussian. The parameters for a conditional Gaussian are shown below.

$$\begin{aligned} p(x|y) &= \mathcal{N}(m_{x|y}, P_{x|y}) \\ m_{x|y} &= \mu_x + \Sigma_{x,y}\Sigma_y^{-1}(y-\mu_y) \quad (10.7) \\ P_{x|y} &= \Sigma_x - \Sigma_{x,y}\Sigma_y^{-1}\Sigma_{y,x}^T \quad (10.8) \end{aligned}$$

The interpretation of these updated parameter values is quite intuitive. For instance, Eq. 10.7 says that the updated mean is the marginal mean of x corrected by some value. This correction term depends on the coupling between the two variables which is encoded in the covariance term,



(a) Joint Gaussian (b) Conditional Gaussian

Figure 10.8. A joint Gaussian distribution of two random variables is shown in figure (a). In (b), we show the result of the distribution over x after conditioning on the value of y.

the original variance term for the observed variable as well as the shift of the observation from the expected value. Notice that if our observation matches the expected value, μ_y, or the covariance between x and y is small, then the correction term is small and thus observing y provides little information about x. Similarly, the covariance is corrected according to the covariance and variance term of the observed variable. Notice here that the correction term is subtracted from the original variance. Since the covariance matrix is positive semi-definite, conditioning on an observed value is guaranteed to decrease variance and therefore reduce our uncertainty about the variable of interest.

Figure 3.2.1 shows an example of two variables that are jointly normally distributed. There is strong correlation between the two variables and thus when we condition on y in figure 10.8(b), the marginal distribution over x changes by shifting (correction to the mean) and scaling (reduction in variance).

3.2.2 Filtering. There are three types of inference we will be interested in computing with the KFM: prediction, filtering, and smoothing. Figure 3.1 shows each of the different procedures, highlighting the variables and connections which are used in each. We will first discuss the filtering problem, updating our parameters of interest upon the arrival of new observations, which subsumes the task of prediction. Then, we will introduce smoothing, estimating parameters given past *and future* observations, which is an offline task that typically provides more accurate estimates with reduced uncertainty.

The objective of filtering is to update our estimates by incorporating the most recent observation. For the KFM, the posterior takes the form

of the Gaussian distribution. This means that to describe our current belief state, x_t, we only need to compute and store the mean vector and covariance matrix. Rewriting the posterior in Eq. 10.9, we derive the functional forms of the distributions to show how we end up with a normally distributed posterior.

$$p(x_t|z_{0:t}) \propto p(z_{0:t}|x_t)p(x_t|z_{0:t-1})$$

From our model assumptions (Eq. 10.4), we have that $p(z_{0:t}|x_t)$ is normally distributed. The second term is the *predicted* belief state given all observations up to the previous time step. This distribution can be rewritten in terms of the model dynamics and a recursion term as shown in Eq. 10.8. We denote the posterior parameters of x from the previous time step as μ_{t-1} and Σ_{t-1}

$$\begin{aligned} p(x_t|z_{0:t-1}) &= \int p(x_t|x_{t-1})p(x_{t-1}|z_{0:t-1})dx_{t-1} \\ &= \int \mathcal{N}(x_t; Ax_{t-1}, Q)\mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}) \\ &= \mathcal{N}(A\mu_{t-1}, A\Sigma_{t-1}A^T + Q) \end{aligned} \tag{10.9}$$

To predict the state at time t, we simply apply the model dynamics to the estimate of the state at $t-1$, integrating over all possibilities. The integration over the previous state is necessary since we are actually uncertain of the true value of x at any given time and thus must consider all possibilities. We use the second term, the posterior of x from the previous time step, weight each guess of the previous state based on our posterior distribution for x_{t-1}. We denote the predicted parameters for x_t as shown in Eq. 10.10 and 10.11.

$$m_t = A\mu_{t-1} \tag{10.10}$$

$$P_t = A\Sigma_{t-1}A^T + Q \tag{10.11}$$

Combining the observation and prior distributions of the belief state (Eq. 10.4 and 10.8), we can reconstruct the joint distribution over x_t and z_t.

$$p(x_t, z_t|z_{t-1}) = \mathcal{N}(\begin{bmatrix} m_t \\ Hm_t \end{bmatrix}, \begin{bmatrix} P_t & P_tH^T \\ HP_t & HP_tH^T + R \end{bmatrix})$$

Algorithm 5 Kalman filtering algorithm

Input: $\mu_{t-1}, \Sigma_{t-1}, z_t$

// predict the current state from previous values
$m_t = A\mu_{t-1}$
$P_t = A\Sigma_{t-1}A^T + Q$
// compute Kalman gain
$K = (P_t H^T)(H * P_t * H^T + R)^{-1}$
// apply correction to predictions using new observation
$\mu_t = m_t + K(z_t - Hm_t)$
$\Sigma_t = (I - KH)P_t$

Conditioning on z_t, we have the measurement update step from 10.7 and 10.8.

$$\begin{aligned} p(x_t|z_{0:t}) &= \mathcal{N}(\mu, \Sigma) \\ K &= HP_t(HP_tH^T + R)^{-1} \\ \mu &= m_t + K(z_t - Hm_t) \quad (10.12) \\ \Sigma &= P_t - K(HP_t)^T \quad (10.13) \end{aligned}$$

Where K is referred to as the Kalman Gain. The inference algorithm for the KFM proceeds by first predicting the $t+1^{st}$ state of x which updates the parameters as described in Eq. 10.8. Then, upon observing a new measurement, z_t, we perform the *measurement correction* step, which updates the parameters for x_t according to equations 10.12 and 10.13. We show the complete filtering inference algorithm for the KFM in algorithm 5 for completeness. From the update equations above, we see how updating belief states upon the arrival of new observations can be computed efficiently, using matrix multiplications and a matrix inverse operation.

3.2.3 Smoothing. Before we get into the smoothing equations for the KFM, we first provide some notation below to identify the different versions of parameters. The first line shows the filtered probability distribution for x_t (explained in the previous section) which we still identify with the parameters μ and Σ. The second line shows the smoothed probability distribution, for which we use the parameters ν and Φ for the mean and covariance to differentiate from the filtered parameters.

$$\begin{aligned} p(x_t|z_{1:t}) &= \mathcal{N}(x_t; \mu_t, \Sigma_t) \\ p(x_t|z_{1:T}) &= \mathcal{N}(x_{t+1}; \nu_{t+1}, \Phi_{t+1}) \end{aligned}$$

Smoothing utilizes observations from the past, present, and future to provide an improved estimate of the belief state. Inference for smoothing

consists of a forward pass through the chain (i.e. filtering) followed by an additional backward recursion in which we consider future observations as well. Specifically, we wish to compute the conditional distribution shown in equation 10.13. Note that in the second step, conditioning on x_{t+1} makes $z_{t+1:T}$ independent of x_t, which is why the extra observations are dropped from this term.

$$\begin{aligned} p(x_t|z_{0:T}) &= \int_{x_{t+1}} p(x_{t+1}, x_t|z_{0:T}) \\ &= \int_{x_{t+1}} p(x_t|x_{t+1}, z_{0:t}) p(x_{t+1}|z_{0:T}) \end{aligned} \tag{10.14}$$

We recognize that $p(x_{t+1}|z_{0:T})$ defines our backward recursion, since this is the smoothed estimate for x_{t+1} given all observations. Therefore, we must derive the parameters for $p(x_t|x_{t+1}, z_{0:t})$ before we can solve the integral in Eq. 10.13. Since we are not given this distribution directly, we will start with the joint distribution over two timesteps, and continue by conditioning on x_{t+1} to get the final distribution.

$$p(x_t, x_{t+1}|z_{0:t}) = \mathcal{N}(\begin{bmatrix} \mu_t \\ A\mu_t \end{bmatrix}, \begin{bmatrix} \Sigma_t & \Sigma_t A^T \\ A\Sigma_t & A\Sigma_t A^T + Q \end{bmatrix})$$

Conditioning on x_{t+1}, we derive the following.

$$\begin{aligned} p(x_t|x_{t+1}, z_{0:t}) &= \mathcal{N}(x_t; m_{t|t+1}, P_{t|t+1}) \\ J &= \Sigma_t A^T (A\Sigma_t A^T + Q)^{-1} \\ m_{t|t+1} &= \mu_t + J(x_{t+1} - A\mu_t) \\ P_{t|t+1} &= \Sigma_t - J\Sigma_t A^T \end{aligned}$$

Plugging these values into equation 10.13, we can now solve the integral.

$$\begin{aligned} p(x_t|z_{0:T}) &= \int_{x_{t+1}} p(x_t|x_{t+1}, z_{0:t}) p(x_{t+1}|z_{0:T}) \\ &= \int_{x_{t+1}} \mathcal{N}(x_t; m_{t|t+1}, P_{t|t+1}) \mathcal{N}(x_{t+1}; \nu_{t+1}, \Phi_{t+1}) \\ &= \mathcal{N}(\mu_t + J(\nu_{t+1} - A), \Sigma_t - JA\Sigma_t) \end{aligned} \tag{10.15}$$

Equation 10.14 shows the final distribution and the parameters for the smoothed estimate of x_t given $z_{0:T}$. The smoothing algorithm works by correcting the filtered estimate of x_t by trying to minimize the difference between the predicted value of the next state and the smoothed estimate for the $(t+1)^{st}$ time step. Finally, the smoothing algorithm initializes the

Algorithm 6 Kalman smoothing algorithm

Input: $\mu_t, \Sigma_t, \nu_{t+1}, \Phi_{t+1}$

// predict state using filtered estimate
$m_+ = A\mu_t$
$P_+ = A\Sigma_t A^T + Q$
// compute Kalman smoother gain
$J = (\Sigma_t A^T)P_+^{-1}$
// apply correction to filtered estimate
$\nu_t = \mu_t + J(\nu_{t+1} - m_+)$
$\Phi_t = \Sigma_t + J(\Phi_{t+1} - P_+)J^T$

smoothed parameters for the x_T to be the same as the filtered estimate for the state at that time, then proceeds backwards through the chain starting at $t = T - 1$ updating each belief state and covariance matrix using the update equations in Eq. 10.16 and 10.17.

$$\nu_t = \mu_t + J(\nu_{t+1} - A\mu_t) \quad (10.16)$$

$$\Phi_t = \Sigma_t - J\Sigma_t A^T \quad (10.17)$$

Lastly, algorithm 6 provides the update algorithm (for a single time step) of the basic Kalman smoother. The algorithm takes as input the filtered parameters of the current time step as well as the smoothed estimates from the $t + 1^{st}$ estimate and produces a smoothed estimate for state t. Similar to the filtering algorithm, we see that the updates are quite efficient, requiring only a few matrix operations.

To conclude this section, we provide a tracking example in figure 3.2.3. The red line represents the true trajectory, the blue points are the observations at each time step. The figure also plots the filtered and smoothed estimates of the trajectory along with a dashed line at a distance of $1\sigma^2$ to represent the estimate uncertainty. It is clear that the filtered trajectory is a substantial improvement over simply connecting the observations. Furthermore, the smoothed estimate improves upon the filtered estimate, resulting in a very close match to the actual trajectory with significantly reduced uncertainty (showing more confidence in the smoothed estimate).

This figure illustrates the importance of applying tracking algorithms when sensors provide noisy data. Simply using the raw sensor data may result in inaccurate trajectories which, if used in a MOD or STDB, will subsequently result in erroneous query results.

Unfortunately, the assumptions upon which the Kalman filter is based (i.e. linear dynamics, Gaussian measurement and system noise) may be too restrictive for some applications and therefore more general techniques are required. In these situations, exact inference becomes in-

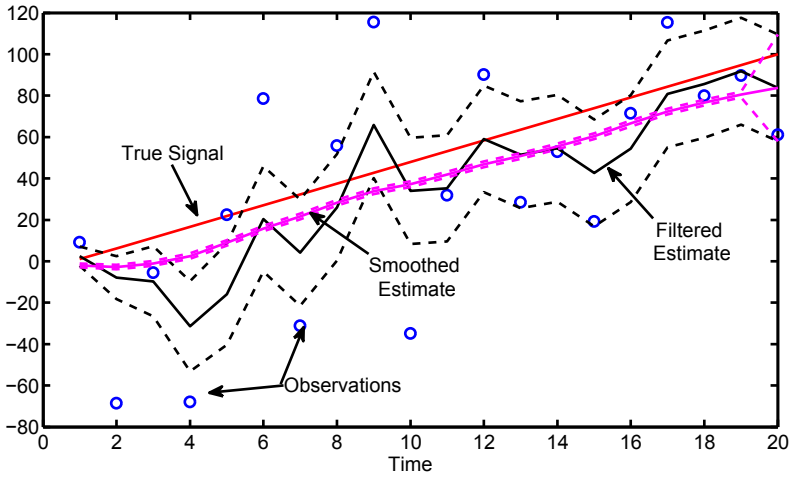


Figure 10.9. An example output of the Kalman filtering and smoothing estimates given a set of noisy observations.

tractable and approximate techniques must be utilized. In the case of non-linear movement, two extensions to the KFM have been introduced which provide local approximations of non-linear movement while (nearly) maintaining the simple filtering update equations. The Extended Kalman Filter (EKF) locally linearizes the state estimation, using partial derivatives of the model dynamics and measurements to approximate updates. Similarly, the Unscented Kalman Filter (UKF) [38] attempts to maintain the efficient update equations of the KFM in the case of non-linear system dynamics by applying the unscented transform, a deterministic sampling technique, to propagate the state distribution through the non-linear dynamics before recovering the parameters of the normal distribution.

Although these approximations provide the convenience of the simple KFM update equations, they typically fail when the dynamics or observation errors result in multi-modal distributions [3]. This is due to the fact that both the EKF and UKF both represent the posterior distribution over x_t as a Gaussian. To represent more complicated densities, a different representation scheme is required as well as approximate inference methods. One of the most popular methods for approximate inference for non-linear dynamics is known as particle filtering [19, 20, 3, 26, 10, 13]. Particle filtering is a generic framework for computing inference in dynamic models in which no special structure exists. The main idea is to represent the probability density function (pdf) describing our belief state of the world as a finite set of weighted point masses. Each

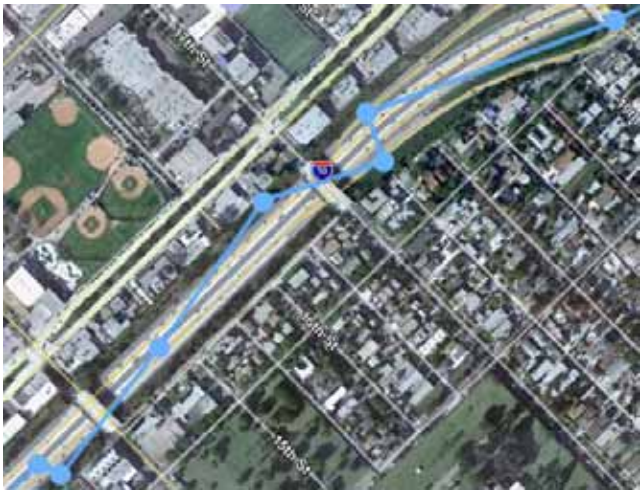


Figure 10.10. An example of the raw trajectory observations overlaid on a (given) road network. Incorporating extra structure, we are able to provide a more accurate path over which the object is likely to have traveled.

point mass is propagated through the model dynamics to predict future states, and is then re-weighted according to the observation likelihood given the predicted value for each. As the number of point masses tends toward infinity, this representation will tend toward the underlying density function and thus can provide a very accurate representation of the system. Several excellent and practical introductions to particle filtering can be found in [20, 10, 3].

3.3 Tracking with Road Networks

Incorporating road network structure into tracking algorithms to improve accuracy has been a popular area of research lately [86, 18, 56, 44]. One such example is the work by Agate and Sullivan [2] in which the authors develop a model for tracking mobile objects that utilizes a road network to constrain object movement and hence improve tracking accuracy. The authors focus on tracking when both ground moving target indicator (GMTI) and high-range resolution (HRR) radar readings are available and thus their observation likelihood models are specific to these measurements. The dynamic model encodes the restriction of the object to only move along the known road segments. Given the road segment upon which the object is currently located, the probability of the next state is a function of the structure of the network since the object is limited to transition to adjacent roads. The state variable

then maintains the current road segment (a discrete ID), a position on that road segment, and a deviation from the road to allow for errors in the road network. Inference for the proposed model is computed using a particle filter which is typical for these complex, non-linear dynamic models. The future position of an object is computed by allowing each particle to take a random walk along the road network for a limited time. Each road segment has a distribution over the amount of time it would take to traverse this segment, thus the point at which the particles stop when time runs out provides a reasonable estimate of the object's next state. The results show substantial improvements over a basic KFM for tracking.

A similar problem is that of map-matching [69], in which an object's noisy position observation is aligned with a known restricted movement surface (e.g. road network). The difficulty in map-matching is the uncertainty in an object's observed location at a given time. Additionally, the road network may be uncertain as well (e.g. user generated maps). Lastly, the problem typically needs to be solved in real-time so the object can identify its true current position and continue navigating to its destination.

A natural model for the task of map-matching is the hidden Markov model (HMM) since it combines information about the distribution over the current state of an object with new (noisy) observations. Newson and Krumm [60] apply an HMM for the map-matching problem using GPS as the observed value and individual road segments as the hidden states. The authors defined the transitions between roads to be based on the distance and connectivity between segments. For instance, a vehicle is more likely to transition to a connected road segment than one that is far away. A similar model is also introduced by Pink and Hummel [67]. In this case, the authors utilize inferred heading information about the vehicle paired with a more accurate representation of the road network based on smooth polynomials instead of linear segments to improve accuracy. Both of these methods rely on consistent and high frequency GPS measurements.

In practice, GPS observations are often obtained irregularly and at low-sampling rates (i.e. 1/120Hz or lower). In these situations, the map-matching problem becomes that of inferring the specific route (sequence of roads traveled) between two GPS observations in an offline setting. Several approaches have been developed specifically for this scenario.

Similar to previous works, Lou et al. [52] address the low sampling frequency map-matching problem by introducing an algorithm that combines a spatial and temporal analysis into an HMM-like model. The Spatio-Temporal-matching (ST-matching) algorithm first attempts to

match each GPS reading to a road segment, then considers the location of the surrounding readings to correct the matched road segment. The underlying assumption behind this approach is that the most direct route is typically the correct one. The temporal analysis utilizes the average speed along each road link. Yuan et al. [97] develop a voting-based map-matching algorithm, called interactive voting map-matching (IVMM), specifically for low sampling rate GPS data. Mapped points are allowed to influence neighboring points with a weight inversely proportional to their distances. The algorithm uses dynamic programming to find the best scoring path given the observations.

Zheng et al. [101] introduce the first data-driven method for resolving the inherent uncertainty of a trajectory collected using a very low GPS sampling rate. The idea is to utilize a collection of historic trajectories and find popular (partial) paths between the sporadic GPS observations. The authors introduce two algorithms for solving the local path problem, one based on greedy-like search process and the other which first extracts a traversal graph containing all of the relevant nodes and edges between two observations and performs a shortest path search in the reduced space. Complete trajectories are then constructed using a dynamic programming algorithm (and a decomposable scoring function). In their experiments, the authors show that their approach significantly outperforms previous methods for dealing with low-sampling-rate trajectories.

Although GPS observations are the most popular type of data for tracking and identifying an object's position, there are other options as well. In fact, continuous collection of GPS can be quite expensive (in terms of power consumption) for a mobile sensor. Therefore, Thiagarajan et al. [79] aim to utilize only the signal from cellular towers, which requires much less energy to collect, to perform map-matching. The authors pose this as a supervised learning problem. In this context the training data is pairs of cellular tower fingerprints (tower IDs and their respective signal strengths) and their corresponding GPS locations (considered to be the labeled data). That is, using the cellular fingerprints as a feature vector, and the GPS location of the user as the actual location, they pose map-matching as a classification problem. Their approach grids the area of interest and uses a HMM to determine the grid after observing the cell tower fingerprint. The authors introduce several additional methods to clean and refine the signal, including integrating information from other sensors on the cell phone (e.g. accelerometer or compass). The experimental results show their method to be a very accurate, energy efficient alternative to constantly using GPS.

Additionally, several other works on map-matching have been introduced that assume spatially and temporally high resolution GPS data is available [6, 7, 25, 34]. These models are similar in that they assume a small degree of error in the observations which allows them to use relatively simple nearest-neighbor approaches to map the object's GPS observation to a road segment on the known road network.

3.4 Tracking for External Sensing

The work mentioned up until this point has all assumed that the mobile objects were providing their location willingly in order to navigate or be queried. However, once this assumption is removed, the problem becomes significantly more challenging. At the core of these challenges is the fact that we do not know which observations belong to which mobile objects, referred to as the *data association* problem. For instance, if two objects are nearby, their observations may get switched, thus the trajectory we obtain would actually be composed of the movements of two different objects.

Although the data association problem makes external sensing a much more complicated problem, it is not the only issue in this scenario. Because sensing occurs in an incredibly noisy environment, we may detect false positives as well as miss the detection of actual objects (false negatives). Moreover, the total number of mobile objects is considered to be unknown. New tracking algorithms have been developed for this scenario, making use of particle filtering methods and finite set statistics (FSS) [42, 53, 61, 64, 86, 85]. The problem scenario of external sensing has not been addressed in the database community, mainly because the current solutions only scale to managing $5 - 10$ objects, as high dimensional filtering is known to be an open problem.

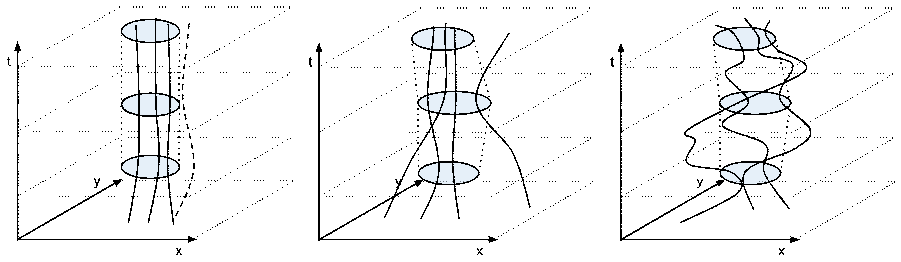
4. Mining Mobility Data

Querying spatiotemporal data is able to provide answers to simple questions, such as *what are the closest coffee shops to me?* or *how many objects have passed through this area over a given time interval*? However, extracting semantically higher-order information from such low level data is a difficult problem. For instance, it may be of interest to identify those mobile objects that behave similarly (e.g. travel to similar locations), identify popular or efficient routes, or just to be able to quantitatively characterize and predict user movements.

We partition this section into three major areas that cover recent work in mining spatiotemporal data: (i) clustering, (ii) route detection, and (iii) movement patterns. The first, covers work on clustering moving

objects or grouping similar trajectories. Clustering is a crucial task in many data management and analysis tasks since it can be used for compression and indexing as well as understanding similarities in the movements of objects. Second, we cover work on popular route detection. This section reviews some recently developed methods for identifying often-traveled paths. The intuition is that the transportation network may not represent all of the factors that influence the decision to take one path over another (e.g. long stop lights, traffic congestion, etc.). By mining previous travel patterns, it is possible to identify the frequently traveled paths. This information can then be used for managing traffic congestion, finding efficient routes, or simply studying the effectiveness of the given road network. The third section focuses on problems related to quantitatively assessing individual user movement patterns. Instead of looking at aggregate behavior, as the work in popular route detection does, the work in this section focuses on the individual. Here the interesting problems are predicting future locations and high-order understanding of user movement (e.g. is the user going to work or to the store?).

Clustering. In the spatiotemporal data setting, clustering aims to group together objects which are within close proximity of one another *and* will remain so over time. Li et al. [47] propose a technique for clustering moving objects by extending the ideas of micro-clustering [99] to handle data that changes over time. To maintain good clusters over time, the authors propose computing a minimum bounding rectangle (MBR) for each cluster. When the MBR reaches a predefined threshold, a split event occurs in which the object furthest from the center of the cluster is removed and reassigned to the nearest microclucster. The resulting time complexity of the clustering approach is $O((N+T)log^2(N+T)log(N))$, where N is the number of mobile objects and T is the total time over which the clustering is to be maintained. Similarly, Jensen et al. [36] also propose an online method for efficiently clustering moving objects based on [99]. The authors introduce a dissimilarity measure for mobile objects which takes the weighted sum of the differences between the locations of two objects over m time steps. The weights are monotonically decreasing as they become further into the future i.e. the current time is weighted more heavily than future positions. Utilizing the BIRCH clustering framework [99], the authors extend the clustering feature vector to include object positions and velocities in a format that is efficient to update. Computing a radius for each cluster (at each time step), future necessary cluster split points can be predicted and then processed efficiently by reassigning.



(a) Flocks (b) Convoys (c) Swarms

Figure 10.11. An example of (a) flocks, (b) convoys, and (c) swarms. Each of these patterns captures groups of objects that tend to move together over time, though they each specify slightly different constraints as highlighted in this figure.

In addition to the basic notions of clustering for mobile objects, newer definitions have also been introduced which provide a stronger notion that objects must *move* together. Recently, the notion of *flocks*, *convoys* and *swarms* have been introduce which impose varying constraints on how tightly packed objects must remain over time. Specifically, in [87], a flock is defined as a set of at least μ trajectories that are located within a given disk with radius $\frac{\epsilon}{2}$, for δ or more time steps. A flock query returns all *sets* of trajectories such that the predicate $flock(\mu, \epsilon, \delta)$ is met. To answer the flock query, the authors propose first griding the space such that each grid is a square with edge lengths ϵ. This reduces the necessary number of comparisons between points and allows the authors to provide an exact answer to the flock query in polynomial time. The authors provide a basic query processing algorithm along with several filtering approaches to improve the algorithm efficiency.

Jeung et a. [37] relax the definition of a flock by using the notion of density connected groups of objects over time. The new spatiotemporal groups are called *convoys* and the authors introduce a filter-and-refine algorithm called *convoy discovery using trajectory simplification* (CuTS) to search for convoys in a given database. The authors first simplify trajectories using linear approximations of subtrajectories such that a maximum error bound is maintained. The simplified trajectories contain fewer points than the originals and are thus easier to manage. The filtering step in CuTS involves a trajectory simplification followed by a density based clustering. In the refinement step, the full trajectories are run through the density clustering again to account for the δ error introduced in the trajectory simplification. The resulting set of clustered trajectories are guaranteed to be convoys.

Furthermore, Li et al. [48, 49] define a *swarm*, which again relaxes the notion of a convoy by removing the restriction that objects must remain

together over consecutive time steps. A swarm is defined as a set of at least min_o objects that remain clustered for at least min_t time steps over a given interval. That is, a swarm is defined by a set of objects, $\mathcal{O}$ and a set of time steps, T, over which the objects remain clustered. To avoid repeatedly identifying subsets of the same objectset, the authors define a *closed swarm* to be a swarm such that the objectset and the timeset are maximal (i.e. adding another object will shrink the timeset and adding another time step will shrink the valid objectset).

Li et al. [48, 49] develop an algorithm for finding swarms which is based on the Apriori (frequent itemset) framework. To manage the exponential search space, three pruning rules are introduced. The first rule says that if $|T| < min_t$, then there is no superset of $\mathcal{O}$ in which the time constraint will be satisfied, thus we can stop growing this set. The second pruning rule, backward pruning, states that if the maximum time set of an objectset, $\mathcal{O}$, does not decrease by adding one more object, then $\mathcal{O}$, may be pruned and we can continue expanding the new objectset $\mathcal{O}' = \{\mathcal{O} \cup o_i\}$. Lastly, forward pruning, by similar means to backward pruning, allows us to determine if an objectset is not closed. Using these pruning rules, the *ObjectGrowth* algorithm performs a depth-first search (on the space of swarms) and identifies all closed swarms.

Popular Route Discovery. Closely related to the problem of clustering, is that of discovering popular routes. Whereas clustering is an object-centric task, the goal of popular route discovery is to identify specific *paths* that are heavily traveled. We review two recent approaches to this problem, the first finds heavily traveled paths conditioned upon a specific origin and destination and the second finds all paths such that the amount of traffic is greater than a given threshold.

Chen et al. [14] introduce a new approach for discovering popular routes only given a set of GPS trajectories. The authors assume that a road network is not available and construct one directly from the data. They use a density based clustering routine for identifying the underlying road network (specifically the intersections) from a set of individual GPS trajectories. The clustering algorithm is an adaptation of DBSCAN [21] with a different connectivity metric which is based on the angle of intersection between trajectories (since roads typically intersect at nearly right angles).

Using the constructed road network, Chen et al. [14] a random walk based approach for identifying popular paths with respect to a specific destination node. The authors assign a transition probability at each intersection by counting the number of objects that took each path from the GPS trajectories. Since the objective is to identify popular routes

between a given source and destination, each trajectory is weighted by the likelihood that it is heading toward the specified destination. That is, separate random walk probabilities are constructed for each specified destination. Using this network, the authors run a random walk using the destination node as an absorbing state to compute a score for each node. The paths are scored by computing the product of the individual node popularity scores (i.e. $\prod_{v \in V} p(v)$) and the path with the maximum popularity is computed using an adaptation of Dijkstra's shortest path search. Although it is not clear if the algorithms will scale to large networks or answering queries online (due to the preprocessing costs), the experimental results presented in [14] are promising.

Similarly, Li et al. [46] propose a method for identifying all of the heavily traveled routes in a given road network, independent of a specific starting and stopping point. The authors introduce a new algorithm, called *FlowScan*, which combines ideas from both individual and aggregate-level analyses over trajectories to identify popular routes. FlowScan iteratively expands a route starting with a given road edge, r, by identifying those edges that are within ϵ hops of r and satisfy the minimum *traffic* support (i.e. number of trajectories that pass through that road segment). The traffic support is computed as $|traffic(r) \cap traffic(s)| \geq MinTraffic$, which says that the two roads must *share* some minimum amount of traffic (i.e. the same trajectories must travel through both road segments). Starting from an edge satisfying the minimum traffic threshold, the algorithm proceeds by adding edges until the conditions are not satisfiable.

The proposed algorithm manages finding popular routes even in difficult instances where routes may partially overlap with one another and there may be sparse regions in which objects may choose from several roads to connect two portions of the same 'route'. The authors experiment with their method on simulated data on a real road network to show the quality of the identified routes obtained by their approach and compare it to prior work. They show that FlowScan is able to identify correct popular routes when the other methods failed, either by grouping overlapping routes together or missing parts of routes due to gaps.

Mobility Patterns. The last group of work we discuss is broadly categorized as mining mobility patterns. By mobility patterns, we refer to the common movements exhibited either by the same object over a long history, or movements repeated by several different objects. For instance, it may be common for residents to use a similar path during their morning commute to get from the suburbs where they live to the downtown area in which they work. Identifying such patterns may be

useful in designing an efficient highway system or estimating road wear over time.

Perhaps the most basic problem involving movement patterns is that of predicting an object's future position given its current location. Tao et al. [76] introduce the *recursive motion function* (RMF) for predicting the future position of a moving object. It formulates an object's location at time t as $\text{loc}_t = \sum_i^f K_m(i)\text{loc}_{t-i}$, where $K_m(i)$ is a constant matrix that represents the type of motion performed by the object and f, called retrospect, is the minimum number of the most recent timestamps which are required to compute the elements of all $K_m(i)$. The $K_m(i)$ matrices represent $m = 1..M$ different motion types (e.g. linear, circular, sine, polynomial, ...) and the motion estimation is done by determining which pattern results in the lowest summed squared distances between the object's actual and predicted trajectory. To manage the large set of potential motion patterns, the authors also propose the spatiotemporal prediction tree (STP-tree), an R-tree based index mobility functions. The STP-tree maintains a set of polynomial curves which represent object movement over time, in the case when the all of the predicted patterns produce linear functions, the STP-tree reduces to the TPR-tree. Although RMFs have performed very well at predicting future locations of mobile objects with complex mobility patterns, they have some weaknesses. Because predictions are based on past movements, RMFs are not able to capture sudden changes in direction (such as a U-turn). Additionally, predictions made several time steps into the future tend to loose accuracy since objects tend to only follow a given motion type for a limited time.

To address these issues, Jeung et al. [37] use previous trajectories from objects to provide a method that is able to accurately predict an object's location multiple time steps in the future. The authors utilize the object's past behavior by performing frequent item-set mining to find common locations (i.e. *given that the object is at the mall at* 4*pm, she will be at the beach at* 5*pm with confidence c*). This work addresses the problem of answering prediction queries by assuming each object has an underlying repetitive pattern. The proposed algorithms are experimentally shown to outperform RMFs, previously the state-of-the-art method for predicting future locations.

Another important aspect of mobility is periodicity as users tend to exhibit many regularities in their movements (e.g. going to work every morning). Li et al. [48, 49] develop a novel technique for identifying periodic movements of a user where the period lengths are also extracted from the data. To robustly identify periodic patterns at various resolutions, the authors propose the idea of using references spots for each

individual. Reference spots are highly visited areas in space which are found using density estimation. Given a reference spot, the movement of a user can be described as a binary sequence in which the user is either at that spot or not. Periodic movement may then be detected by applying a Discrete Fourier Transform (DFT) and selecting all frequencies higher than a threshold. This procedure is repeated for each reference location, thus identifying various period lengths which are robust to noise in the spatial movement of the user.

Periodic movement patterns are then described as a probability distribution (computed via maximum likelihood) over reference locations at each time step within a given period. Using these probability distributions, the next step is to identify the specific patterns. This is accomplished through a hierarchical clustering of the probability distributions, using KL-divergence as the distance measure. Patterns are combined and an overall clustering score is maintained such that when the score increases too much, the clustering stops and hence picks k, the number of clusters, automatically. Experimental evaluation found the proposed methods to be able to accurately identify interesting periodic movement behaviors. Additionally, the authors applied their method to a real data set, the location of a bald eagle over three years, and they were able to identify the migration patterns of the eagle over that time.

In addition to utilizing the large quantities of available trajectory data to identify interesting patterns, it is also possible to identify object movements that *do not* follow the expected behavior. That is, given enough data we can identify common behavior patterns and use this to detect anomalies. To this end, Li et al. [46] introduce a rule and motif based anomaly detection method for moving objects (ROAM). The idea in this work is to partition trajectories into several prototypical submovements and use features extracted from these movements to classify each trajectory as normal or abnormal. In this work, the authors pose anomaly detection as a supervised learning problem and thus assume a training set of labeled trajectories is available. The proposed algorithm is composed of three steps: (i) motif extraction, (ii) feature generation and (iii) classification.

The motif extraction phase slides a window (of fixed length) over each trajectory and the set of subtrajectories (i.e. each window) is clustered. The cluster centroids are referred to as *motifs*, and a second pass through the dataset determines which trajectories *express* each motif (i.e. have a subtrajectory that matches with error less than ϵ). Next, features are generated for each motif and the values are discretized (or clustered) to ensure generalization (e.g. (right-turn, speed, 11mph) or (right-turn, time, 2am)). Additionally, a hierarchy is built over the value space

for each specific feature (e.g. 'right-turn speed') to provide a multi-resolution view of the data. Lastly, to utilize this feature hierarchy, the authors propose classification using hierarchical prediction rules (CHIP) which is based on FOIL [70]. CHIP learns a set of rules by exploring features in a top down manner, such that it will first attempt to classify based on high level features in the hierarchy. To determine if it should expand a lower feature level, CHIP computes the information gain from using the higher resolution features. The experiments show that the proposed technique significantly outperforms a basic SVM classification. In fact, ROAM is shown to consistently outperform competing methods as both the number of trajectories and the number of motifs increases.

Combining several of these ideas, [8, 50] build high-order models of user movements in which subtrajectories may be identified as completing specific tasks. By high-order modeling, we refer to the ability of a model to assign semantically meaningful labels to segments of a trajectory (e.g. "going to work") and include more abstract attributes (e.g. mode of transportation) over the raw trajectory data. In this work, the objective is to be able to answer queries *not* about the specific trajectory, but about the purpose of the movement (e.g. given a trajectory is it more likely that the subject is going to work or the grocery store?). Specifically, Liao et al. [50] introduce a dynamic Bayesian network model which incorporates high level goals, such as "going to work" or "going to the supermarket" into the model which may be inferred from low level location data (e.g. GPS). The result is a model which is capable of querying a variety of aspects of a user's movement. The authors show that their proposed model can be learned in a completely unsupervised manner, though applying the semantic categories such as "going to work" must be supervised, and is able to identify locations of interest and abnormal behavior in a real data trace.

In addition to mining patterns of *movement*, a related problem is to identify those *locations* that are visited by a large number of trajectories. Using the abundant amount of GPS trace data available over a region, it is possible to find specific locations that are of interest (e.g. Statue of Liberty, good restaurants, popular bars, etc.) [9, 84, 102]. For instance, Zheng et al. [102] develop a PageRank-like algorithm for mining interesting locations by considering each user's travel experience. The basic idea is that users that are well traveled within a region of interest will likely know more of the relevant locations and thus a visit from one of these *travel authorities* should be weighted more than a visit from a tourist who does not know the local area well. Alternatively, Uddin et al. [84] identify regions of interest (ROIs) from trajectory data by looking for dense areas (at least N mobile objects in a fixed area) in which mobile

objects spend some minimum amount of time. The authors index trajectories by velocity, then extend the Pointwise Dense Region (PDR) [62] method to identify ROIs.

Similarly, the work by Giannotti et al. [22] combine aspects of mining interesting locations, with understanding and predicting movement patterns. The authors extend prior work on mining frequent patterns and define a spatiotemporal sequence (ST-sequence), which is a sequence of locations visited by a set of users with a given level of support. The goal in this work, much like in the frequent itemset mining, is to identify a frequently visited sequence of locations over time. The authors address the problem when the locations, or regions-of-interest (ROI), are known as well as when they must be extracted from the data. More recent work has built upon these concepts of pattern mining in the spatiotemporal domain as well [58, 72].

5. Discussion and Future Research Directions

Over the past two decades, we have seen significant advancements in all areas related to managing spatiotemporal data. In this chapter, we attempt to cover the state-of-the-art solutions to the current challenges specific to managing spatiotemporal data. Our review focused on data management techniques as these are fundamental to nearly all applications involving mobility data. Additionally, we also covered some of the core and recent work on tracking mobile objects. Lastly, we introduced some of the recent applications of mining spatiotemporal data to extract interesting patterns.

Despite the multitude of work in these areas, new and challenging problems are constantly being introduced. Below we briefly outline a few potentially interesting areas of future research.

- **Combining spatiotemporal information with social networks:** Work on social network analysis in the recent years has been plentiful due to the explosion of the availability of social data from sites such as Facebook, Twitter, MySpace, and various other relationship or communication networks. The analysis of users and their connections has largely focused on the concept of homophily, which is the tendency of individuals to connect to others that are similar to themselves. However, physical space is another significant factor which influences how users interact with one another. Combining information about a user's movement with her social network presents an exciting research direction [16, 96].

- **Data-driven Techniques:** Large quantities of spatiotemporal data are becoming readily available through several research ef-

forts and sharing sites [106–108]. For example, the California Department of Transportation has made the data it collects from highway loop counters publicly available (for free) [107]. Utilizing these data stores and developing data-driven techniques to tackle core problems such as map-matching [101], identifying locations of interest [102] or traffic analysis [33] continues to be a promising area of research.

- **Communication Efficiency:** Communication between mobile objects, as well as between mobile objects and a central database, remains an expensive operation in terms of power and bandwidth consumption. Making use of low quality sensors which use less power, or scheduling the transfer of data in a more effective manner can both help to reduce these costs [79, 83].

- **Information Movement:** In this chapter, we have focused on managing the mobility data of physical objects traveling through space, however, studying the flow of information offers a similar set of challenges. With the growth of the internet, there has been an astronomical increase in the availability and sharing of information. Only recently have researchers started to ask questions about how information gets disseminated over time [1, 23, 45, 93]. Additionally, combining the challenges invovled in tracking information and moving objects, which are used to transfer information, results in yet another set of interesting problems known as *data ferrying* [103].

Acknowledgements

This work was partially supported by the National Science Foundation under grant IIS-0917149.

References

[1] E. Adar and L. Adamic. Tracking information epidemics in blogspace. In *Web Intelligence*, pages 207–214. Ieee, 2005.

[2] C. S. Agate and K. J. Sullivan. Road-Constrained Target Tracking and Identification Using a Particle Filter. In *SPIE*, number 805, 2003.

[3] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[4] P. Bakalov, M. Hadjieleftheriou, and V. Tsotras. Time relaxed spatiotemporal trajectory joins. In *GIS*, pages 182–191. ACM, 2005.

[5] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. 19(2), 1990.

[6] M. Bierlaire, J. Chen, and J. Newman. A Probabilistic Map Matching Method for Smartphone GPS data. Technical report, EPFL, 2010.

[7] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *VLDB*, pages 853–864, 2005.

[8] H. H. Bui. A General Model for Online Probabilistic Plan Recognition. In *IJCAI*, 2003.

[9] X. Cao, G. Cong, and C. Jensen. Mining significant semantic locations from gps data. *Proceedings of the VLDB Endowment*, 3(1-2):1009–1020, 2010.

[10] O. Cappe, S. J. Godsill, and E. Moulines. An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, May 2007.

[11] V. P. Chakka, A. C. Everspaugh, and J. M. Patel. Indexing Large Trajectory Data Sets With SETI. In *CIDR*, 2003.

[12] S. Chen, C. S. Jensen, and D. Lin. A Benchmark for Evaluating Moving Object Indexes. In *PVLDB*, pages 1574–1585, 2008.

[13] Z. Chen. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. Technical report, Adaptive Systems Lab, McMaster University, Hamilton, Ontario, 2003.

[14] Z. Chen, H. T. Shen, and X. Zhou. Discovering Popular Routes from Trajectories. In *ICDE*, 2011.

[15] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying Imprecise Data in Moving Object Environments. *TKDE*, 16(9):1112–1127, 2004.

[16] E. Cho, S. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *SIGKDD*, pages 1082–1090. ACM, 2011.

[17] B. S. E. Chung, W.-C. Lee, and A. L. P. Chen. Processing probabilistic spatio-temporal range queries over moving objects with uncertainty. In *EDBT*. ACM Press, 2009.

[18] A. Civilis, C. S. Jensen, and S. Pakalnis. Techniques for Efficient Road-Network-Based Tracking of Moving Objects. *IEEE Trans-*

actions on Knowledge and Data Engineering, 17(5):698–712, May 2005.

[19] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.

[20] A. Doucet and A. M. Johansen. A Tutorial on Particle Filtering and Smoothing : Fifteen years later. Technical Report December, 2008.

[21] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, volume 1996, pages 226–231. AAAI Press, 1996.

[22] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Trajectory Pattern Mining. In *KDD*, pages 330–339, 2007.

[23] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):21, 2012.

[24] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *VLDB*, pages 794–805. VLDB Endowment, 2007.

[25] J. S. Greenfeld. Matching GPS Observations to Locations on a Digital Map. *Environmental Engineering*, 1(3):164–173, 2002.

[26] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, 2002.

[27] R. H. Guting, T. Behr, and J. Xu. Efficient k-nearest neighbor search on moving object trajectories. *The VLDB Journal*, 19(5):687–714, Apr. 2010.

[28] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD Conference*, 1984.

[29] M. Hadjieleftheriou, G. Kollios, P. Bakalov, and V. J. Tsotras. Complex spatio-temporal pattern queries. In *VLDB*, 2005.

[30] P. Hoff. *A first course in Bayesian statistical methods.* Springer Verlag, 2009.

[31] J. Hosbond, S. Saltenis, and R. Ortoft. Indexing uncertainty of continuously moving objects. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, pages 911–915. IEEE, 2003.

[32] M. Hua and J. Pei. Probabilistic Path Queries in Road Networks: Traffic Uncertainty Aware Path Selection. In *EDBT*, 2010.

[33] A. Ihler, J. Hutchins, and P. Smyth. Adaptive Event Detection with Time-Varying Poisson Processes. In *KDD*, 2006.

[34] A. Jawad and K. Kersting. Kernelized Map Matching for Noisy Trajectories. In *SIG SPATIAL*, 2010.

[35] C. Jensen, D. Lin, and B. Ooi. Query and update efficient b+-tree based indexing of moving objects. In *VLDB*, pages 768–779. VLDB Endowment, 2004.

[36] C. Jensen, D. Lin, and B. Ooi. Continuous clustering of moving objects. *Knowledge and Data Engineering, IEEE Transactions on*, 19(9):1161–1174, 2007.

[37] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A Hybrid Prediction Model for Moving Objects. In *ICDE*. IEEE, Apr. 2008.

[38] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, volume 3, page 26. Spie Bellingham, WA, 1997.

[39] R. Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[40] B. Kanagal and A. Deshpande. Online Filtering, Smoothing and Probabilistic Modeling of Streaming data. In *ICDE*, 2008.

[41] U. Khan and J. Moura. Distributing the kalman filter for large-scale systems. *Signal Processing*, 56(10):4919–4935, 2008.

[42] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE transactions on pattern analysis and machine intelligence*, 27(11):1805–19, Nov. 2005.

[43] K.-S. Kim, S.-W. Kim, T.-W. Kim, and K.-J. Li. Fast indexing and updating method for moving objects on road networks. In *Web Information Systems Engineering Workshops*, pages 34–42. Ieee, 2003.

[44] W. Koch. On Bayesian Tracking of Extended Objects. In *Multisensor Fusion and Integration for Intelligent Systems*, pages 209–216. Ieee, Sept. 2006.

[45] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *SIGKDD*, pages 497–506. ACM, 2009.

[46] X. Li, J. Han, J. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. *Advances in Spatial and Temporal Databases*, pages 441–459, 2007.

[47] Y. Li, J. Han, and J. Yang. Clustering Moving Objects. In *KDD*, 2004.

[48] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. *VLDB Endowment*, 3(1-2):723–734, 2010.

[49] Z. Li, J. Han, M. Ji, L. Tang, Y. Yu, B. Ding, J. Lee, and R. Kays. Movemine: Mining moving object data for discovery of animal movement patterns. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4):37, 2011.

[50] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, Apr. 2007.

[51] H. Loose, U. Franke, and C. Stiller. Kalman particle filter for lane recognition on rural roads. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 60–65. IEEE, 2009.

[52] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *GIS*, number c. ACM Press, 2009.

[53] R. P. S. Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1152–1178, Oct. 2003.

[54] N. Malviya, S. Madden, and A. Bhattacharya. A Continuous Query System for Dynamic Route Planning. In *ICDE*, 2011.

[55] Y. Manolopoulos, A. Nanopoulos, and Y. Theodoridis. *R-trees: Theory and Applications*. Springer-Verlag New York Inc, 2006.

[56] O. Mazhelis. Using Recursive Bayesian Estimation for Matching GPS Measurements to Imperfect Road Network Data. In *Intelligent Transportation Systems*, pages 1492–1497, 2010.

[57] M. F. Mokbel, T. M. Ghanem, and W. G. Aref. Spatio-temporal Access Methods. *IEEE Data Engineering Bulletin*, 2010.

[58] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *SIGKDD*, pages 637–646. ACM, 2009.

[59] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference, and Learning*. Phd, University of California, Berkeley, 1994.

[60] P. Newson and J. Krumm. Hidden Markov Map Matching Through Noise and Sparseness. In *SIG SPATIAL*, 2009.

[61] W. Ng, J. Li, S. J. Godsill, and S. K. Pang. Multitarget Initiation, Tracking and Termination Using Bayesian Monte Carlo Methods. *The Computer Journal*, 50(6):674–693, Sept. 2007.

[62] J. Ni and C. Ravishankar. Pointwise-dense region queries in spatio-temporal databases. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1066–1075. IEEE, 2007.

[63] E. Nikolova, M. Brand, and D. R. Karger. Optimal Route Planning under Uncertainty. In *International Conference on Automated Planning and Scheduling*, 2006.

[64] S. K. Pang, J. Li, and S. J. Godsill. Detection and Tracking of Coordinated Groups. *IEEE Transactions On Aerospace And Electronic Systems*, 47(1), 2009.

[65] J. Patel, Y. Chen, and V. Chakka. Stripes: an efficient index for predicted trajectories. In *SIGMOD*, pages 635–646. ACM, 2004.

[66] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel Approaches to the Indexing of Moving Object Trajectories. In *VLDB*, pages 395–406, 2000.

[67] O. Pink and B. Hummel. A statistical approach to map matching using road network geometry , topology and vehicular motion constraints. In *Intelligent Transportation Systems*, pages 862–867, 2008.

[68] S. Prabhakar, Y. Xia, D. Kalashnikov, W. Aref, and S. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Transactions on Computers*, 51(10):1124–1140, 2002.

[69] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C*, 15(5):312–328, Oct. 2007.

[70] J. Quinlan and R. Cameron-Jones. Foil: A midterm report. In *ECML*, pages 1–20. Springer, 1993.

[71] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD*, pages 331–342, June 2000.

[72] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. Campbell. Nextplace: a spatio-temporal prediction framework for pervasive systems. *Pervasive Computing*, pages 152–169, 2011.

[73] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: a Dynamic Index for Multi-Dimensional Objects. In *VLDB*, 1987.

[74] D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *Control Theory and Applications, IET*, 4:1303 – 1318, 2010.

[75] Z. Song and N. Roussopoulos. Seb-tree: An approach to index continuously moving objects. In *Mobile Data Management*, pages 340–344. Springer, 2003.

[76] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD*. ACM Press, 2004.

[77] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB*, pages 287–298. VLDB Endowment, 2002.

[78] Y. Tao, D. Papadias, and J. Sun. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In *VLDB*, 2003.

[79] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *Networked Systems Design and Implementation (NSDI)*, 2011.

[80] G. Trajcevski. Probabilistic range queries in moving objects databases with uncertainty. In *MobiDE*, pages 39–45. ACM Press, 2003.

[81] G. Trajcevski, R. Tamassia, P. Scheuermann, and I. F. Cruz. Continuous Probabilistic Nearest-Neighbor Queries for Uncertain Trajectories. In *EDBT*, 2009.

[82] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems (TODS)*, 29(3):463–507, 2004.

[83] H. Tsai, D. Yang, and M. Chen. Mining group movement patterns for tracking moving objects efficiently. *Knowledge and Data Engineering, IEEE Transactions on*, 23(2):266–281, 2011.

[84] M. Uddin, C. Ravishankar, and V. Tsotras. Finding regions of interest from trajectory data. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 39–48. IEEE, 2011.

[85] M. Ulmke, F. Fkie, and P. Willett. Gaussian Mixture Cardinalized PHD Filter for Ground Moving Target Tracking. In *Information Fusion*, number 3, 2007.

[86] M. Ulmke and W. Koch. Road-map Assisted Ground Moving Target Tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 42(4):1264–1274, Oct. 2006.

[87] M. Vieira, P. Bakalov, and V. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *SIGSPATIAL*, pages 286–295. ACM, 2009.

[88] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 2006.

[89] D. Wilkie, J. V. D. Berg, M. Lin, and D. Manocha. Self-Aware Traffic Route Planning. In *Artificial Intelligence*, pages 1521–1527, 2011.

[90] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 588–596. IEEE, 1998.

[91] S. Won, W. Melek, F. Golnaraghi, et al. A kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system. *Industrial Electronics*, 57(5):1787–1798, 2010.

[92] X. Xiong and W. Aref. R-trees with update memos. In *ICDE*, pages 22–22. IEEE, 2006.

[93] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *ICDM*, pages 599–608. IEEE, 2010.

[94] J. Yim, J. Joo, and C. Park. A kalman filter updating method for the indoor moving object database. *Expert Systems with Applications*, 38(12):15075 – 15083, 2011.

[95] M. Yiu, Y. Tao, and N. Mamoulis. The b dual-tree: indexing moving objects by space filling curves in the dual space. *The VLDB Journal*, 17(3):379–400, 2008.

[96] X. Yu, A. Pan, L. Tang, Z. Li, and J. Han. Geo-friends recommendation in gps-based cyber-physical social network. In *ASONAM*, 2011.

[97] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An Interactive-Voting Based Map Matching Algorithm. In *Mobile Data Management*, pages 43–52, 2010.

[98] M. Zhang, S. Chen, and C. S. Jensen. Effectively Indexing Uncertain Moving Objects for Predictive Queries. In *VLDB*, 2009.

[99] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.

[100] K. Zheng, G. Trajcevski, X. Zhou, and P. Scheuermann. Probabilistic range queries for uncertain trajectories on road networks. In *EDBT*, pages 283–294. ACM, 2011.

[101] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing Uncertainty of Low-Sampling-Rate Trajectories. In *ICDE*, 2011.

[102] Y. Zheng, L. Zhang, X. Xie, and W.-y. Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *WWW*, pages 791–800, 2009.

[103] Y. Zhu, W. Wu, and V. Leung. Energy-efficient tree-based message ferrying routing schemes for wireless sensor networks. *Mobile Networks and Applications*, 16(1):58–70, 2011.

[104] `http://www.gstatic.com/ads/research/en/2011\_TheMobileMovement.pdf`.

[105] `http://www.idc.com/getdoc.jsp?containerId=prUS23112511`.

[106] `http://www.openstreetmap.org/`.

[107] `http://pems.dot.ca.gov/`.

[108] `http://www.movebank.org/`.

Chapter 11

A SURVEY OF RFID DATA PROCESSING

Charu C. Aggarwal
IBM T. J. Watson Research Center
Hawthorne, NY 10532, USA
charu@us.ibm.com

Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL, USA
hanj@cs.uiuc.edu

Abstract

Radio Frequency Identification (RFID) is a new technology which allows a sensor (reader) to read, from a distance, and without line of sight, a unique product identification code (EPC) associated with a tag. Such tags are very useful in inventory management and logistics, because they can be used in order to track the movement and locations of large volumes of items in a cost effective way. This leads to massive streams of noisy data, which can be used in the context of a variety of data management and event processing algorithms. The use of RFID also has a number of privacy challenges associated with it, because a tag on an item being carried by a person, also becomes a unique location tag for that person. Therefore, methods need to to be designed to increase the privacy and security of RFID technology. This chapter will provide a broad overview and survey of a variety of RFID data management, mining and processing techniques. We will also discuss the privacy and security issues associated with the use of RFID technology.

Keywords: RFID Data, RFID Mining

1. Introduction

RFID technology is a recent sensor technology, which allows *uniquely identifiable* tags to be read from a distance with the use of a sensor reader. RFID sensor technology is useful for tracking *very large volumes of items with specific identifiability* in a cost effective way. When combined with more sophisticated sensors transmitting real-time information and internet-enabled web services, this allows real-time connectivity and tracking of information about a wide variety of objects in daily life. The capability has been recognized in sensor computing as a paradigm shift in how objects are tracked in a ubiquitous manner, and is generally referred to as the *Internet of Things* [8].

At the most basic level, the definition of Radio Frequency Identification (RFID) is as follows: *RFID is a technology which allows a sensor (reader) to read, from a distance, and without line of sight, a unique product identification code (EPC) associated with a tag* [34]. Thus, the unique code from the tag is transmitted to one or more sensor reader(s), which in turn, transmit(s) the readings to one or more server(s). The data at the server is aggregated in order to track all the different product codes which are associated with the tags. *Passive RFID Tags* do not require an onboard battery, and can typically be read from a distance of a few centimeters to a few meters. Passive tags are typically powered by the radio signal that reads them. On the other hand, *active tags* come equipped with an onboard battery, which provides larger read ranges. If the tags are equipped with a sufficiently powerful antenna, it is also possible for them to transmit very long range signals, such as enabling the readability of the signal from satellites. While active tags have larger ranges, they come at a larger unit cost, and also have limited life spans. For most retail applications, passive tags are used in order to minimize the costs associated with the infrastructure. The primary fixed cost of such an infrastructure is embedded in the hardware and software associated with the sensor readers, whereas the variable costs of this system are associated with the RFID tags, each of which needs to be affixed to a tracked item. Typically, the number of sensor readers being used in large scale applications (such as retail tracking) is relatively small compared to the number of objects being tracked. For example, in a typical retail application, each tracking point will have a small number of sensor readers, which keep track of a very large volume of RFID tags passing through that point.

RFID sensors vary from conventional sensor technology in a variety of ways. With most sensors (whether mobile or stationary), the objects or readings which are sensed are not done so *actively*, and are usually not

powered by a sensor reader. Conventional sensors are used in conjunction with a battery for tracking (and wireless transmission of) readings such as ambient sound, temperature, light, videos, pressure, locations, or other objects, which are then transmitted by the sensor itself. In the case of RFID, the key is to identify specific information in the tag, by powering it with a sensor reader. This specific information in the tag is also known as *Electronic Product Identification Code (EPC)*. The uniqueness of identification code, and the cost-effectiveness of the tag, allows the simultaneous tracking of a large number of objects, since the presence of an object at a particular location can be associated with the identification code on its tag.

While the core idea of RFID technology is not new, and dates back to World War II for distinguishing between friendly and enemy aircrafts [53, 50], recent years have seen the emergence of a new *stripped down* version of the tag, which lacks a power source or antenna, and does little more than provide a unique identifier. Unlike regular sensors, such tags are *extremely inexpensive*, cost no more than a *few cents* each, and can easily be constructed for large scale applications. Some of the earliest discussions on the the rapid advancement of RFID technology to such large scale applications may be found in [56, 71]. The trend towards, *smaller*, *unobtrusive*, and *inexpensive* tags is exemplified by the following:

- Zebra has developed a print engine, which can embed an RFID transponder directly into a product label [16].

- Hitachi has developed an extremely tiny RFID tag, known as the μ-chip, which can be directly embedded into photocopier paper [81]. This can be used for document tracking.

These different kinds of developments suggest the continuing miniaturization of RFID technology across different domains. Furthermore, these developments also suggest that the applications of RFID technology go well beyond retail applications. It is important to note that the complexity of an RFID tag can be fairly flexible, depending upon the problem domain. If desired, it is possible to incorporate sensing into RFID technology [71, 72] with the use of onboard sensors that generate data dynamically. For example, an RFID tag may incorporate a temperature sensor (for perishable goods), or a passive force sensor, which can return information about the possible damage to a product, if it is dropped. Such tags are typically *active* RFID tags (with an onboard battery), and they are typically more expensive than *passive* tags, which are powered by a sensor reader, and return only the EPC. The par-

ticular choice of the tag depends upon the application domain, and an acceptable price-point for the tag in that application domain.

Thus, the broad flexibility in the functionality of RFID tags, makes them widely applicable to different problem domains. Examples of such domains are as follows:

- **Retail Applications:** In retail applications, RFID tags are associated with the products, and fixed sensor readers at particular locations are used in order to track the movement of products. The technique can be used for real-time inventory tracking. Alternatively, active shelves can be used in order to determine product availability.

- **The Internet of Things:** Ubiquitous computing, which is also referred to as the *internet of things*, has been identified as a key trend in recent years, in which information about objects is continuously tracked with the use of sensor technology. Along with a host of other advances in embedded sensor technology, RFID has been identified to be one of the key enabling technologies towards this trend [74]. The application of RFID technology for enabling such ubiquitous computing requires the coupling of basic RFID technology with the relevant web and internet services for allowing ubiquitous tracking. In particular, the ability to simultaneously identify a large number of objects uniquely in a cost-effective way with RFID tags has been a driving force in this direction.

- **Medical Applications:** RFID has increasingly found acceptance in a variety of pervasive healthcare applications [67]. For example, the tags may be associated with the patient medical history. This can be useful for automated tracking of patient medical history. For RFID-enabled healthcare asset management, major healthcare equipments, such as wheelchairs or other medical equipments are RFID-tagged, so that health-care experts can locate any asset in real-time. This can also help increase emergency room safety in addition to time saving [68].

- **Payment Systems:** RFID tags are used as credit-card like payment tokens that contain a serial number. When the tag is scanned for a payment, the reader transmits the number over a network to a remote computer, which is authorized to debit the money from the consumer's back account. An example of such a payment system is Texas Instruments's *Speedpass*, pay-at-pump system, which was introduced in Mobil stations in the mid-nineties.

- **Access Control:** The transmitters which are mounted on vehicles emit a signal which is read by the sensors at the tolls. This is used to keep track of the number of accesses through the toll, and also control the access of the vehicle through the toll. A popular such system is the New York's *EZ–Pass*, which is equipped with a 921.75 MHz semi-passive tag. Access control to commercial buildings and installations for employees and workers is often managed with the use of cards with RFID chips embedded in them.
- **Animal Movement:** Animals can be implanted with RFID tags in order to trace their movement. Alternatively, pet owner information can also be implanted on the RFID tag. Both kinds of tags can be useful in locating a lost pet, or in tracking the patterns of movement of wild life. More sophisticated tags (equipped with GPS receivers and transmitters) have been used in order to transmit signals that can be picked up by satellite, and have been used to track aquatic animals and other wild life.
- **Library Tracking:** RFID technology can be be used in order to automate the tracking of items which are checked out from the library by patrons. Such RFID tags also serve as security devices with the use of exit sensors, which track items that are being removed from the library, but have not been checked out. For example, several libraries such as the Santa Clara City library in California, the University of Nevada, Las Vegas library, and the Eugene, Oregon public library have already tagged every book, tape, CD, or other item in their collection [55].
- **Airline Luggage Management:** In this case, the RFID chips are attached to the luggage tags. Therefore, by placing sensor readers at strategic locations, it is possible to track the movement of luggage. This has been shown to be very useful in reducing lost luggage [54, 66].
- **Automobile Immobilizers:** Some of the newer car models have keys which contain an RFID tag. This key is authenticated by the steering column, and is required for vehicle operation. Such immobilizers typically have a small read range of a few centimeters, and operate in the low frequency end of the electromagnetic spectrum. Such systems have been widely credited with greatly reducing auto theft.

Even though RFID technology has been around for many years, its use for *large scale applications*, has only recently found widespread acceptance. The massive nature of RFID data is associated with numerous

challenges from the perspective of mining and analysis. These challenges are as follows:

- The volume of data associated with RFID can be extremely large, because of the large number of tags which may be tracked by a single reader. Furthermore, in some applications, the number of readers may also be quite large, which leads to a *high fan-in and hierarchical* organization of the underlying sensor network [23]. Such a system poses numerous challenges, because successive hierarchical aggregation of streams from different nodes of the network can lead to a overwhelming amount of data at the higher level nodes. It has been argued in [23] that a uniform stream-oriented query processing approach at all levels of the hierarchy works best in such systems.

- The data can be very *noisy* and *redundant*, with many tags being completely dropped, and others being read by multiple readers at multiple instants, resulting in tremendous redundancy of the representation. Furthermore, the large volume of the data makes the process of cleaning much more challenging. Therefore, effective methods need to be designed to compress and clean such data. The cleaning process can be rather expensive, and challenging, especially when near real-time responses to location queries are required.

- Many applications such as high level semantic event detection can be extremely challenging because of the high volume of the stream, and the real time nature of such applications. The noise and errors in the underlying data can lead to additional ambiguities during the event detection process.

- RFID deployments lead to a number of privacy concerns, because tags are uniquely identifiable by readers. Therefore, by carrying a tag attached to clothing, it may be possible to covertly track people without their knowledge. A variety of methods need to be designed in order to increase the privacy and security aspects of RFID technology.

- As with any sensor infrastructure, RFID technology and readers vulnerable to partial or complete system failures. Such failures can also lead to challenges in data processing, because data which is not collected will always be missing from the database. If the missing data is not explicitly accounted for by the underlying data analytics, it may lead to inaccurate inferences, because the missing

data may be interpreted as the absence of a particular item, rather than a system failure.

The chapter will discuss the processes involved in the storage, management and cleaning of RFID data. The chapter is organized as follows. In the next section, we will discuss the process of RFID Data Cleaning and compression. In section 3, we will discuss issues in the data management and warehousing of RFID data. An important goal of tracking RFID data is to use it for detecting interesting semantic events in the data, especially in real-time streaming scenarios. Section 4 discusses methods for event detection from RFID data streams. Section 5 discusses issues related to privacy and security of RFID data. Section 6 contains the conclusions and summary.

2. Raw RFID Data Cleaning and Compression

A variety of middleware architectures are used in order to collect and process RFID data [11, 22, 33, 37, 80, 68]. RFID data, by its very nature, is extremely noisy, incomplete and redundant because of cross-reads from multiple sensor readers. For example, it has been shown [38, 65] that a large fraction of the readings in RFID streams are essentially dropped. It has been estimated in [38, 65], that as many as 30% of the sensor readings are lost (i.e. the tag identifiers do not appear at all), because of the reader unreliability. Therefore, RFID middleware systems are used in order to correct for dropped readings. A commonly used method in many data cleaning systems [33, 80] is to use a temporal smoothing filter, in which a sliding window over the reader's data stream interpolates for lost readings from each tag within the time window. This approach provides each tag more opportunities to be read within the smoothing window. This reduces the number of distinct tags which are lost, because they will show up in one or more tag readings, when the window size is increased. Typically, the window size is fixed, as in [22], and the smoothing is performed on the basis of the readings which are received within this fixed window.

It has been observed in [36], that the choice of window size can be a critical parameter, which leads to different tradeoffs between false positives and false negatives. Using a window size which is too small will lead to missed readings (or *false negatives*), because the tag has fewer opportunities to be scanned by the reader. On the other hand, a larger window size will cause *false positives*, because it will lead to scanned readings, even after the tag has moved out of the reader's detection range. The work in [36] proposes *SMURF (Statistical sMoothing for Unreliable RFid data)*, which is an adaptive smoothing filter for raw RFID data streams.

This technique determines the most effective window size automatically, and continuously changes it over the course of the RFID stream, depending upon the underlying readings. One characteristic of this approach is that it does not expose the smoothing window parameter to the particular application at hand. This makes the approach much more flexible in different scenarios.

The approach proposed in [36] views RFID readings as *unequal probability random sample* of tags in the physical world. Therefore, the tradeoff between reader unreliability and tag dynamics can be explored in a principled and statistical manner. Furthermore, the approach can be used to clean both "single-tag" and "multiple-tag" readings. In the multiple tag case, it is assumed that single readings do not need to be tracked. For example, a store may only need to track when the number of items of a particular type falls below a given threshold. For the single tag case, binomial sampling methods are used for the cleaning process. For the multi-tag case, the aggregate signal over a tag population is cleaned with the use of Horvitz-Thompson estimators.

One characteristic of effective cleaning methods is to use *declarative methods* in the cleaning process [38, 39, 36]. The broad idea is to specify cleaning stages with the use of high-level declarative queries over relational data streams. Once this is done, the system can translate the queries into the required low level operations. Such an approach is useful in helping programmers avoid writing low level interaction code, by specifying the queries at the high level. Furthermore, such an approach makes the system data- and device-independent, and the code does not need to be changed if the underlying device fails, or is upgraded.

We note that the middleware approach to RFID data cleaning performs all the processing on the data upfront, before applying any of the data querying or analytical methods on it. However, different applications may define the anamolies or corrections on the same data set in a different way. Therefore, the method in [59] introduces a deferred approach for detecting and correcting RFID anamolies. Each application uses declarative sequence-based rules in order specify, detect, and correct relevant anamolies. We note that this approach is generally different from the methods proposed in [22, 36], which make the cleansing process application-independent. Clearly, both approaches have their own advantages in different scenarios. The generally accepted principle [22] is that the separation of the middleware from the applications is a desirable goal, because of the diversity of the applications in which such data could be used, and the network limitations of the underlying readers.

The actual process of cleaning may be much more complicated in an application containing thousands of readers and millions of tags. In such a case, the process of cleaning may incur tremendous *costs* associated with the entire process. These costs may be associated with either the cleaning plan itself, or in the misclassification associated with the cleaned data records. Therefore, a method for cost-conscious cleaning of massive RFID data sets has been proposed in [27].

The work in [27] assumes that three different kinds of inputs are available:

- A set of tag readings are available, which form a representative sample of the possible set of readings. Each reading is associated with a correct location of the tag, contextual information, area conditions, and tag protocol.
- A set of cleaning methods with associated per-tuple cleaning costs are specified.
- A per-tuple mis-classification cost is specified, which may be constant, a function of the tag reading and incorrectly assigned location.

The goal of the cost-sensitive approach is to learn a cleaning plan that identifies the conditions (feature values) under which a specific cleaning method or a sequence of cleaning methods should be applied in order to minimize the expected cleaning costs, including error costs. The work in [27] proposes a cleaning method which dynamically adjusts the probability of tag-presence based on the last observation. This is essentially a *Dynamic Bayesian Network (DBN)* approach. It has been shown in [27] that such an approach can outperform or complement methods which are based on smoothing windows. One advantage of DBN-based cleaning is that it does not require the use of recent tag readings (as in a window-based method), and it also gives more importance to recent readings, since the probability of tag-presence is continuously adjusted by the incoming tag readings.

A method called *StreamClean* has been proposed in [46], which uses *global integrity constraints* in order to clean the data. The core idea in *StreamClean* is that the tuples in a data stream system are not random, but are often related to one another, according to application-specific criteria. An example of such an integrity constraint provided in [46] can be as follows:
A car parked in the garage at time $t_e < t$ must either have exited in (t_e, t), or it must still be parked at time t.
In essence, the approach in *StreamClean* requires the specification of

user-stated properties that are true about the data. The system then uses these properties in order to insert missing tuples or correct conflicting tuples. In the event of groups of conflicting tuples, a probability of correctness is assigned to each tuple. Thus, the *StreamClean* approach transform the data to a probabilistic representation, in which explicit probability values are assigned to tuples. The approach then transforms constraints on the tuples into constraints on the underlying probability values. This also allows the possibility of *soft constraints*, in which a probability of a fact being correct is specified, rather than a *hard constraint*, in which the fact is deterministically known to be correct. The *StreamClean* method uses a non-linear optimization method, where the objective is to determine a probability assignment that maximizes entropy while satisfying the integrity constraints. The intuition behind maximizing entropy [32] is that in the absence of additional knowledge, the underlying solutions should be as uniform as possible. For example, the use of entropy maximization results in the explicit assumption, that in the absence of stated constraints, the probabilities of different input tuples are independent of each other. While this may not necessarily be true in all solutions, it is the most reasonable assumption to make in the absence of prior beliefs about such tuples.

It has been observed in [29] that RFID data exhibits a considerable amount of redundancy because of multiple scans of the same item, even when it is stationary at a given location. In practice, one needs to track only interesting movements and activities on the item. This is an issue which we will discuss in some detail in the next section on data management and warehousing. RFID tag readings also exhibit a considerable amount of *spatial redundancy* because of scans of the same object from the RFID readers placed in multiple zones. This is primarily because of the spatial overlap in the range of different sensor readers. This provides seemingly inconsistent readings because of the inconsistent (virtual) locations reported by the different sensors scanning the same object. It has been observed in [15] that the redundancy is both a blessing and a curse. While the redundancy causes inconsistent readings, it also provides useful information about the location of an object in cases, where the intended reader fails to perform its intended function. In addition, it has been observed in [15], that a considerable amount of background information is often available, which can be used in order to enhance accuracy. This background information is as follows:

- Prior knowledge about tagged objects and readers can be used in order to improve accuracy.

- Information about the constraints in the underlying application, such as the maximum capacity of a room or shelf, can be used in order to improve accuracy.

The work in [15] proposes a Bayesian inference framework, which takes full advantage of the duplicate readings, and the additional background information in order to maximize the accuracy of RFID data collection.

A different method, proposed in [52], is to use a *KerneL dEnsity-bAsed Probability cleaning method (KLEAP)* to remove cross-reads within a sliding window. The method estimates the density of each tag using a kernel-based function. The idea is that the most relevant reader will have a much larger number of objects in a similar position as this object. Therefore, the reader corresponding to the micro-cluster with the largest density will be regarded as the relevant position of the tagged object in the current window. The reads which are derived from the other readers will be treated as cross-reads.

3. RFID Data Management and Warehousing

Some of the earliest work on temporal management of RFID data was proposed in [68]. This work develops a *Dynamic Relationship ER (DRER) Model* for temporal management of RFID data. This system is built on top of the ER model with relatively few extensions. The technique maintains the history of events and state changes, so that complex queries can be supported. A rules-based framework is used to transform business logic data into user configured rules. In addition to location, another concept which is introduced is that of *containment*. Containment implies a hierarchical relationship between a set of objects. For example, a pallet may be loaded with cases, and both the pallet and the cases would have their own separate EPCs.

The RFID data contains two basic categories of data, corresponding to *static* and *dynamic* data. The static data is related to commercial entities such as location information, product level information, and serial information. There are two kinds of dynamic data: (a) The first corresponds to *instance data* such as serial number and the date of manufacture; and (b) The second corresponds to *temporal data* such as location observations and temporal changes in the containment of objects.

The second kind of temporal data are captured through EPC tag readings, and is related to the movement of products. The four primary kinds of entities which interact with one another in such a system are *EPC-tagged objects*, *readers*, *locations*, and *transactions*. These entities interact with one another, as object locations change, and entity containment relationships change as well. We note that even sensor (reader)

locations may change over time, as they are moved from one place to the other. Besides state changes, events are also generated in the interactions, including *observations*, when EPC tags interact with readers, and *transacted items*, when an object participates in a transaction.

The dynamic entity-relation model (DRER) is an extension of the ER model. In the ER model, all entities and relationships are static or current. In the RFID system, entities are static, but the relationships between them are dynamic. Thus, the only addition to the traditional ER model is the addition of a new kind of relationship, known as the *dynamic relationship*. There are two kinds of dynamic relationships, one of which generates events, and the other generates state history. An event-based dynamic relationship is associated with a single attribute known as `timestamp`, which represents the time at which the event occurred. On the other hand, a state-based dynamic relationship is associated with two attributes `tstart` and `tend` corresponding to when the state started and ended.

Thus, in the DRER model, we have three different static entities corresponding to *sensor reader*, *object*, and *location*. In addition, an entity called *transaction* may be used in order to represent business transactions, though we omit it in the discussion here for simplicity. Each of the static entities is associated with its own set of static entity tables. State-based dynamic relationships correspond to *sensor location*, *object location*, and *containment*. We note that each of these relationships are dynamic, and naturally have a starting and ending time. Event-based dynamic relationships occur at a particular instant, and may correspond to an *observation*, which is generated by a sensor reading an EPC tag. The different static and dynamic tables in the DRER model, together with their attributes are summarized below:

Entity	Type	Table Attributes
Sensor	Static	`SENSOR(sensor_epc, name, description)`
Object	Static	`OBJECT(object_epc, name, description)`
Location	Static	`LOCATION(location_id, name, owner)`
Observation	Dyn.	`OBSERVATION(sensor_epc, value, timestamp)`
Containment	Dyn.	`CONTAINMENT(epc, parent_epc, tstart, tend)`
Obj. Location	Dyn.	`OBJECTLOCATION(epc, location_id, tstart, tend)`
Sens. Location	Dyn.	`SENSORLOCATION(sensor_epc, location_id, position, tstart, tend)`

These tables can be used in conjunction with a variety of SQL queries in order to resolve interesting aspects about the RFID objects. Some examples [68] of such queries are as follows:

RFID Object Tracking Queries: The `OBJECTLOCATION` table carries

most of the information needed for formulating RFID location tracking queries. Since the starting and ending time for each object is included in the table, a query can be performed on an EPC in order to determine the history of locations for that objects. The precise time taken for an object to move from one location to another can also be derived because of the presence of the `tstart` and `tend` variables. This history of locations can be sorted temporally in order to provide the history of locations for any object. Similarly, missing objects at a location can be obtained from the `OBJECTLOCATION` table by comparing the objects at that location, with the set of objects at any time and location, where they were previously known to be complete. The precise formulation of these queries is provided in [68].

RFID Data Monitoring Queries: It is also formulate containment or observation queries for any particular snapshot by making use of the `CONTAINMENT`, `OBJECTLOCATION` and `OBSERVATION` queries. In addition temporal joins can be performed between different objects by formulating queries which examine the overlap between their `tstart` and `tend` variables. For example, recursive containment can be easily queried with this approach with the use of `CONTAINMENT` table, and temporal aggregation can be performed on the number of items which passed through a location at a given time, by making use of the `tstart` and `tend` attributes of the `OBJECTLOCATION` table. Thus, the tables supported by the DRER model are *expressive* and can support a wide range of SQL queries [68].

We note that in order to transform the noisy RFID into the high level semantic tables discussed above, which are consistent and non-redundant, a number of rules need to be defined. These rules correspond to *data filtering*, *location transformation*, and *data aggregation*. We note that many relationship tables (such as *containment* tables) are not *explicitly* specified in the RFID data, and they need to be inferred and aggregated, based on the observation patterns. A rule-based framework is proposed in [68] in order to automate the transformation of primitive events into semantically cleaner representations. For example, a *data filtering* rule can be defined to scan the data within a sliding window in order to determine if there are duplicates of the same event in multiple readers. One of these can then be dropped. Similarly, when a new location for an object is defined by a particular reader, the ending time stamp for the last location is updated to the current time. An entry is created in the `OBJECTLOCATION` table which a new starting time stamp, which is the current time. The ending time-stamp for the new entry is set to `UC (Until Changed)`. This is an example of a *location transformation rule*. An example of a *data aggregation* rule is one in which

when a set of pallets are loaded onto a track, the set of EPC readings for all the objects are inserted as the children of the EPC of the truck, in the `CONTAINMENT` table. Thus, an event detector continuously monitors the observation streams, and triggers actions which generate the corresponding data.

One challenge with managing RFID data, which was noticed in [68] was that RFID data typically have very large volume, which can lead to accumulation of large volumes of data. This can lead to slower queries and updates. An important observation about RFID data is that they typically have a limited life span, starting from the time it is first tagged, to the time when it is sold to the customer. Therefore, the database management approach in [68] partitions the data into an *active* set of RFID data, which corresponds to items which are frequently updated; and an inactive set of data, which corresponds to items that are no longer updated frequently. Since the majority of the data becomes inactive over time, this leads to much faster queries of the active data during its life-cycle.

3.1 Efficient Warehousing of RFID Data

A related, but somewhat different kind of RFID data management and warehousing has been discussed in [29]. This approach is designed towards finding the relevant paths of items in the RFID scenario. This process is also designed towards modeling the dynamic relationships such as *containment*, except that it does so not just for explicit containment, but also for items which move together. Also, the mapping relationships are modeled somewhat differently. The approach is also designed for tracking specific *measures* associated with the RFID items, which is typical in a data warehouse.

As in the case of [68], methods need to be designed to handle the massive redundancy of different types. These could be because of multiple readings of the same item from the same reader at multiple times. Consider the situation, where a typical reading from an RFID tag is of the form $(EPC, Location, Time)$. We note that the same tag may be read many times at the same location, even though no significant event may have occurred involving the time. As in [68], the only two readings which are significant are the first and last moment at which the items were read. The work in [29] uses two main kinds of compression.

- **Temporal Compression:** Multiple scans of the same code at the same (virtual) location can be compressed significantly. For example, if an item is loaded on an ship from one port to another, then the virtual location of the item corresponds to "*ship*" and all scans

of the item aboard the ship can be reduced to two scans. In practice, the location is associated with the identifier of a fixed sensor reader with a virtual location, such as "*ship*". These two scans correspond to the time that the item was loaded on the ship, and the time at which the item was removed from the ship. Therefore, by storing the times when the item was first moved to the vicinity of the reader and the time that it was moved away from the reader, all the relevant information is represented [29]. This allows us to represent the item in the form $(EPC, Location, TimeIn, TimeOut)$. We note that the $TimeIn$ and $TimeOut$ variables are similar to the `tstart` and `tend` variables proposed in [68] for maintaining the object location tables.

- **Group-based Compression:** In most real scenarios, RFID items often move together in groups or consignments. For example, all items which are loaded onto the ship stay together throughout the trip. Therefore, all the individual RFID of the items can be replaced by a single *generalized identifier*, or GID. In practice, groups of items may split or merge, as items are loaded at ports from different sources, or split into different destinations. Correspondingly, the generalized identifiers can be arranged hierarchically, in order to effectively represent these merges and splits.

One challenge with the use of RFID data with traditional data warehousing techniques, is that traditional warehousing methods do not properly consider the spatial links between different data records, which are essential in the RFID scenario. Therefore, traditional dare warehousing techniques may fail, when they are directly applied to RFID data. For example, consider the situation, where the cleaned RFID representation is of the form $(EPC, Location, TimeIn, TimeOut : Measure)$, where "Measure" could correspond to a value such as the quantity of the item present at the given location. Such a representation could be used in order to respond to queries such as the number of items which are present at a given location at any given period. However, it cannot be used to determine the number of items which moved from one location to another in a given period, at least with traditional data warehousing operations.

Therefore, RFID warehouses can be represented in the form of three different tables [29]: (a) an *info* table which contains location independent information about the items, such as its SKU, Product type etc., (b) a *stay* table which essentially contains all the set of facts in the form $(EPC, Location, TimeIn, TimeOut : Measure)$ (or in aggregate form as GIDs instead of EPCs), and (c) a *map* table which contains the links

between the different records of the fact table. The map table links together the different records in the table which form a path.

We note that the map table is the only additional information which needs to be maintained in the case of an RFID data warehouse. The RFID warehouse can be viewed as a multi-level database, in which the lowest level of representation are the raw RFID records, whereas the higher levels contain the cleansed and compressed records. In addition to the use of group-wise movements for compressing the data, a variety of other abstractions can be used for further compression. For example, if a minimum time granularity of one hour is required, then the set of movements and stays occurring in a single hour can be consolidated into a single movement. Similarly, the location can be specified at a higher level of granularity, and the sizes and the types of products can also be consolidated. This is because users are often interested in queries at much higher abstraction levels. Many path segments which are less important can also be eliminated and consolidated into a single movement. For example, for a store manager, the movement of items between shelves may not be important, and can either be eliminated or consolidated with some other movement. All of these operations significantly reduce the size of the representation, and make higher level query processing operations much more efficient.

Some other characteristics of the different kinds of tables such as the *info table*, *stay table*, and *map* table are as follows:

- The *info* table contains path-independent dimensions. Each dimension can have an associated concept hierarchy on which OLAP operations can be performed. For example, one could drill down on a particular product category and support aggregate queries on this category.

- The *stay* table contains the *TimeIn* and *TimeOut* information for the different products. In order to save space, this information is stored in terms of aggregated GIDs of items which move together, rather than the individual EPC values.

- The map table contains the hierarchy of GIDs in the data. Each entry is of the form $(gid, (gid_1 \ldots gid_r))$. This implies that gid points to $gid_1 \ldots gid_r$. We note that at the lower levels, gid_k could correspond to an individual EPC. The higher levels of the gid, are also labeled with locations, with one identifier corresponding to each location for items in the gid.

We note that the use of the gids, as maintained by the mapping table can provide a very efficient way to perform the queries, since each

individual gid may contain a very large number of items which have traveled together. In order to materialize the measures such as counts the algorithm does not need to access the counts of the individual EPCs.

It has been observed in [30] that the movement trails of RFID data form gigantic commodity flowgraphs, which represent the locations and durations of the path stages traversed by each item. The work in [30] proposes a method to construct a warehouse of commodity flows, which is also referred to as a *FlowCube*. Similar to OLAP, the model comprises cuboids, which aggregate the item flows at a given abstraction level. In this case, the measure of each cell is a commodity flowgraph, which captures the major movements and trends, as well as significant deviations from the trend in each cell. The flowgraph can also be viewed at multiple levels by changing the abstraction levels of path stages. The latter is achieved by performing simultaneous aggregation of paths to all interesting abstraction levels, In addition, path segments with low frequency and rarely occurring cells are removed from the representation.

It has been observed in [28] that a clustered path database, which is natural to RFID applications, can be naturally modeled as a *compressed probabilistic workflow*. Each location corresponds to an activity, and locations are linked according to their order of occurrence. A link between two activities has a probability, which represents the percentage of time that one of the activities occurred immediately after the other. A probabilistic representation of the workflow can also be used in the context of the FlowCube. The details of such a concrete probabilistic workflow are provided in [28].

4. Semantic Event Extraction from RFID Data Streams

The discussion so far has focussed on low level cleaning, event extraction and data management of RFID. However, in many applications, the events to be discovered are *high level semantic events*, as opposed to the primitive event of an object moving from one location to another. Such events are also referred to as *complex* events. The problem of event mining in RFID processing is related to previous research on complex event detection in active databases and high fan in sensor systems [1, 7, 13, 14, 18, 26, 62, 76, 70, 79]. In particular, the work in [62] discusses a high fan-in architecture for a sensor network, and shows how it can be used in order to process complex events by combining RFID data with other kinds of sensor readings and stored data.

An example of such a high-level semantic event discussed in [78] is the shoplifting example, in which the event corresponds to an item be-

ing picked up at a shelf and then being taken out of the store without being checked out. Clearly, a sequence of occurrence or non-occurrence of primitive RFID events can be used to determine the occurrence of a higher level semantic event. The problem of complex event extraction is probably one of the most critical ones in RFID event processing, because the purpose of tracking RFID data is essentially to determine different kinds of semantic events based on application-specific criteria. Therefore, an expressive and user-friendly language is required to support this class of queries for event processing. A language called *SASE* was proposed in [78] for complex event processing over RFID data streams.

The *SASE* event language is a declarative language that combines *filtering*, *correlation* and *transformation* of simpler events, in order to determine how they are correlated on the basis of time- and value constraints. The *SASE* language uses the following form in order to determine events:

```
EVENT <event pattern>
[WHERE <qualification>]
[WITHIN <window>]
```

For example, the shoplifting event pattern can be captured using the following construct [78]:

```
EVENT SEQ(SHELF-READING x, !  (COUNTER-READING y),
                  EXIT-READING z)
WHERE x.id = y.id ∩ x.id = z.id
WITHIN 12 hours
```

We note that the `EVENT` clause of the above contains a `SEQ` construct, which specifies a set of (primitive) events in a particular order. In this case, the construct detects a `SHELF` reading, followed by *the absence* of a `COUNTER` reading, and then followed by an `EXIT` reading. The `SEQ` construct turns out to be quite useful in the context of a wide variety of RFID queries, because of its ability to detect sequential combinations of basic events. Such sequential combinations form the core of event detection in complex RFID scenarios.

The basic constructs such as `SEQ` and negation are already available in the existing languages. However, in the context of RFID data, a number of new features are added by the work in [78], such as the use of parameterized predicates for correlating events via value-bed constraints. Sliding windows are used for imposing temporal constraints. Methods are also proposed for resolving the semantic subtlety of negation, when used together with sliding windows.

A query plan in the *SASE* language uses a subset of the following six operators in order to resolve the queries:

- **Sequence Scan and Construction (SSC):** While sequence scan and construction are technically different operators, they are always used together. The corresponding component is referred to as *SSC*. When a query contains the `SEQ` construct in the language, all the *positive* components of the `SEQ` specification are handled by the sequence scan and construction operators. Thus, a subsequence of the original specification is handled by this pair of constructs. Thus, the function of the *SSC* is to transform a stream of events into a stream of event sequences, each of which represents a unique match of specified *SSC* sub-sequence type.

- **Selection:** This is the commonly used operator in relational query processing. In this case, this operator is used to filter each event sequence by applying different predicates.

- **Window:** The window operator imposes the constraint of the `WITHIN` clause. For each event sequence, it checks if the temporal difference between the first and last events is less than the specified window T.

- **Negation:** The negation operator handles the negative components of a `SEQ` construct which have been ignored by SSC.

- **Transformation:** This operator converts each event sequence to a composite event by concatenating attributes of all the events in the sequence.

Another recent method for event processing with RFID data has been proposed in [9]. This method has the ability to query different readers for data in order to make key real time inferences for events. In addition, methods have been designed to work with the code embedded in the RFID tags for event processing. The EPC tags represents a string, in which different portions of the string correspond to different parts of the information about the product. Therefore, any algorithm needs to be able to work effectively in terms of deciphering the importance of different portions of the string for event processing. The approach in [9] shows how to extend an SQL-based query language in order to make it suitable for event processing in the context of RFID data. This can be an advantage in many scenarios, because users are often more familiar with SQL-like languages. Because of this, recent systems for event processing [19] have generally tried to work with extensions of the SQL language for event processing.

It has been observed [6] that stream event detection algorithms can be generally formulated as *pattern matching algorithms* over data streams.

Many of the traditional database operators for stream processing cannot effectively handle the detection of arbitrary event patterns. These techniques are quite effective for regular expression matching, though not quite as effective for pattern matching. The latter is much more important in the stream scenario. Therefore, the work in [6] proposes a method for matching arbitrary patterns in data streams in order to perform event detection. This work proposes a formal query evaluation model, NFA^b that combines a finite automaton with a match buffer. This is used to create query evaluation plans that can be executed over event streams. One characteristic of this approach is that it uses storage sharing of all possible pattern matches as well as in automaton execution to produce these matches. This results in more efficient query execution.

4.1 Probabilistic Event Extraction

It has been observed in [45, 47] that there is an inherent ambiguity in the cleaning and determination of high level events of RFID data. Since, the collection of RFID data is prone to errors, it is natural that such data is best represented by probabilistic databases as discussed in [17, 31, 77]. The importance of using probabilistic representations for event extraction in pervasive computing applications has been discussed in depth in [25], though the approach discusses the design of an *inference engine* for event extraction. In the context of RFID data management applications, it is more critical to design a *query processing engine* for probabilistic event extraction. Therefore, the work in [45] proposes a probabilistic event language *PeexL* for defining probabilistic events. An implementation of the approach is proposed in a system called Probabilistic Event EXtractor *(PEEX)*, a middleware layer on top of a relational database management system (RDBMS). The idea is that uncertainty propagates as events are aggregated into higher level events. For example, a `MEETING` event can be inferred from a sequence of `ENTERED-ROOM` events by different participants. However, if there is limited confidence in the `ENTERED-ROOM` events, then the confidence in the `MEETING` events will also be lower. The work in [45, 47] uses confidence tables in order to track the confidence of the different events and then aggregate these probabilities into higher level event probabilities with the use of the *PeexL* language. Another interesting probabilistic event processing system known as *Lahar* has been proposed in [61]. This approach uses a framework which is similar to the *Cayuga* system [18] for event processing, except that it is focussed on querying probabilistic representations of the underlying data.

5. Privacy and Security Issues with RFID Data

One challenge with the use of RFID technology is that the tags on the items can be tracked by sensor readers without the knowledge or consent of people carrying them. For example, the items bought in a store can be used in order to track people, as they move about in the world. This is particularly true for items such as shoes or clothing. This has lead to increasing privacy concerns about the large-scale use of such technology [24, 43, 51, 57]. For example, the *Consumers Against Supermarket Privacy Invasion and Numbering (CASPIAN)* protested against apparel manufacturer Benetton for planning to attach RFID tags to their products. This lead to a boycott of those products in 2003 [57, 82]. *CASPIAN* similarly criticized Tesco for conducting experimental trials of tags on a variety of its products [83] in 2005.

An additional troubling aspect of the tags is that they contain no information about their read-history. While tags can be scanned by anyone without the consumer's knowledge, there is also no way for the consumer to know that they have been scanned. The EPC contains a serial number, which is unique to a particular *instance of the* product item. Therefore, once a customer buys the product and carries it on their person, the product EPC becomes a unique identifier for the customer, which can be distinguished from a similar product bought by another customer. This information can be misused in a variety of ways:

- Individuals carrying tagged products can be tracked with the use of covert readers placed at different locations.
- Since the EPC also contains manufacturer information, it can be used in order to order to obtain competitive information about customer preferences without their knowledge.
- When tagged items move from one individual to the other, the transactions between different individuals can be tracked.
- Associations are often built up between tagged items and individuals in corporate information systems, as individuals move around with tagged items over time. When these items are discarded, such associations are typically not broken. If these items are then used for malicious or illegal purposes, then this can expose the individual to different kinds of liabilities with law enforcement.

In addition to the personal privacy threats, a number of threats are possible with the use of RFID data at the corporate level. A particular area of concern is the tracking of RFID data for the purposes of *corporate espionage*. Tagged objects in the supply chain make it easy

for competitors to routinely gather information about the activities of a business.

The use of RFID technology also has security consequences which go beyond simple privacy concerns. These are as follows:

- RFID technology is highly dependent on the use of radio signals which are easily jammed. This can open the system to a variety of infrastructure threats.
- It has recently been demonstrated [10], that RFID tags can be cloned to emit the same identification code as another tag. This opens the system to fraud, when the RFID tag is used for the purpose of sensitive tasks such as payment. This can also be used in order to make the function of an automobile immobilizer vulnerable to attack.

We note that privacy issues for RFID data can arise both during *data collection* and during *data management*, once the RFID data has been captured. For the case of data collection, the information is typically stolen through eavesdropping on either the tag or the reader signal. In this case, since the privacy concerns arise from the design of the tag itself, many of the issues need to be addressed by enhancement and modification of the underlying tag, with either hardware or software solutions, or a combination of both. On the other hand, in the case of data management, the privacy issues relate to the access control of the underlying data. We will discuss some of the different methods for privacy preservation both during data collection and management in the following subsections.

5.1 The Kill Command

The Auto-Id Center designed the "kill" command, which are intended to be executed at the point of sale. The kill command can be triggered by a signal, which explicitly disables the tag [63, 64]. If desired, a short 8-bit password can be included with the "kill" command. The tag is subsequently "dead" and no longer emits the EPC, which is needed to identify it. However, the killing of a tag, can sometimes be an impractical solution in cases, where the tags have a utility beyond the point of sale. Some examples are as follows:

- The tags are used for identification purposes in order to facilitate the repairs or returns for the underlying products.
- Many smart appliances use the tags for other purposes. An example discussed in [24] discusses the smart refrigerator which uses

RFID tags in order to identify expired food. Clearly, the killing of a tag at the point of sale would make this functionality useless.

Therefore, a number of methods have been proposed, which go beyond the "kill" command for the purposes of providing privacy protection. A slightly softer solution is to use a locking and unlocking mechanism for the tags [75]. For example, the tags could be locked at the time of check out in the store. The tags can then be unlocked with a meta-id provided by the consumer, along with an associated PIN. This approach has two primary disadvantages. One disadvantage is that the incorporation of smart technology makes the tag much more expensive. The second disadvantage is that it is impractical for consumers to manage meta-identifiers and PINs for all the different products that they may buy.

5.2 Cryptographic Solutions

A possible solution is to encrypt the code in a tag before transmission. However, such a solution may not be very effective, because this only protects the *content* of the tag, but not the ability to *uniquely identify* the tag. For example, the encoded tag is itself a kind of meta-tag, which can be used for the purposes of tracking. Another solution is to embed dynamic encryption ability within the tag. Such a solution, however, comes at a cost, because it requires the chip to have the ability to perform such an encryption computation. Another solution which has recently been proposed [40] is to perform the cryptographic computations at the reader end itself, and store the resulting information in the tags. This solution of course requires careful modification of the reader-tag protocols. A number of cryptographic protocols for privacy protection of library RFID activity are discussed in [55]. Some of the cryptographic schemes [44, 48, 58] work with *re-writable* memory in the tags in order to increase security. The tags are encrypted, and the reader is able to decrypt them when they send them to the server, in order to determine the unique meta-information in the tag. The reader also has the capability to re-encrypt the tag with a different key and write it to its memory, so that the (encrypted) tag signal for an eavesdropper is different at different times. Such a scheme provides additional protection because of repeated change in the encrypted representation of the tag, and prevents the eavesdropper from uniquely identifying the tag at different times.

5.3 Blocker Tags

An interesting solution for making it difficult to read tags in an unauthorized way is the use of *blocker tags* [41, 42]. Blocker tags exploit the collision properties of RFID transmission, which are inherent in this technology. The key idea is that when two RFID tags transmit distinct signals to a reader at the same time, a broadcast collision occurs, which prevents the reader from deciphering either response. Such collisions are in fact very likely to occur during the normal operation of the RFID infrastructure. In order to handle this issue, RFID readers typically use anti-collision protocols. The purpose of blocker tags is to emit signals (or spam) which can defeat these anti-collision protocols, thereby causing the reader to stall. The idea is that blocker tags should be implemented in a way, that it will only spam unauthorized readers, thereby allowing the authorized readers to behave normally.

Typically the anti-collision protocols which are used are also referred to as *singulation* protocols, which allow the tag reader to systematically explore all the tags in a certain order with the use of a *tree-walking protocol*, which singles out all the tags for scanning in a specific order. This is achieved by treating the binary code on each tag in the form of a binary tree, where each node in the tree is considered a prefix of the binary tree. The idea is that the reader has the capability to scan for tags containing only a particular prefix, and ask all other tags to remain "silent". Tags which contain that particular prefix, transmit their bit which comes just after that prefix. The algorithm starts at the root of the tree, and scans the first bit of the tags. In the event that both 0 and 1 is transmitted, then a collision will occur, which is detected. This means that both branches of the tree need to be explored, since there are tags which contain both a 0 and a 1 in the first. Clearly, a collision is quite likely to occur at the higher levels of the tree. On the other hand, if only a 0 is transmitted, then the left branch of the tree needs to be explored. Otherwise, the right branch of the tree is explored. This process is used to recursively traverse the portion of the tree which is relevant to the RFID tags being scanned. This recursive traversal finally reaches the leaves of the tree, at which point, the tags are recorded uniquely by the reader. It is clear that for a 96-bit Class 1 EPC tag, the portion of the tree which is explored by the reader is an extremely tiny fraction of the 2^{96} possible nodes in the tree, since the number of distinct tags being present would be much smaller than 2^{96}. In fact, the entire tree-size is too large to be explored by the tree-walking algorithm.

The blocker tag takes advantage of this property and forces (malicious) readers to explore the full tree of size 2^k, which would cause the

reader to stall. The idea is that for each query by the reader, the blocker tag *always* sends *both* and 0 and a 1. This means that the reader would have to recursively traverse the entire tree, an undesirable situation, which could cause the reader to stall. The blocker tag can be modified to block only those tags with certain prefixes, a process which is referred to as *selective blocking*. For example, the blocker tag may respond to the reader only for those prefixes which correspond to the left subtree of the root. This means that all RFIDs which start with a "0" are now protected or "blocked" from scanning. Thus, by carefully assigning prefixes to different products, it is possible to selectively block only certain kinds of products for the purposes of privacy protection. Alternatively, it may be possible to reset the prefix in a tag at check-out time, so as to move the tag from the unprotected zone to the protected zone, once it has been bought by the consumer. A blocker tag may either be carried by a consumer on their person (through active acquisition), or may be provided by a supermarket in a grocery bag, so as to prevent undesired scanning of the items bought by a consumer. Once the items are removed from the bag (for example, when food items are placed in a refrigerator), the tags can become usable again, since they have been removed from the vicinity of the blocker tag. The main drawback of blocker tags is that they only provide an "opt-out" mechanism, in which tags are active by default, and consumers must take the step of acquiring blockers in order to protect their privacy. Such opt-out mechanisms are very useful, when the tags only need to be blocked at certain times, places, or in the possession of certain people.

We further note that a *polite blocking* protocol can be implemented, which allows the readers to query the blocker tags, which tells them the portions of the tree that they should not traverse [41]. Thus, the blocker tag is being "polite" to the reader in telling it, which portions of the tree it is blocking. The tree-walking protocol can then be modified in order to not query those portions of the tree which are being blocked. Polite blocking is useful, when the environment may contain legitimate readers, which should not be made to inadvertently stall by the use of blocker tags. Since authorized readers are likely to follow the proper protocol, they will not be affected by the blocker tag. Furthermore, even if unauthorized readers use the proper protocol, they will be unable to access the tags of items with protected prefixes. This is the entire purpose of blocking.

The blocking approach can be considered a kind of passive jamming, and can be used both for privacy protection or for malicious purposes. When used for malicious purposes, it can be considered equivalent to a

"denial of service" attack, which prevents readers from performing their normal function by jamming them.

5.4 Other Privacy- and Security-Protection Methods

A number of privacy-protection mechanisms rely on the fact that the eavesdroppers are more likely to be at some distance from the tag. In this context, it was inferred by [75] that the greater threat to privacy arises from the eavesdropping of signals sent from the reader (which can be detected much further away), rather than reading the tag itself (which can be done only at a much closer distance). In fact, the IDs being read by the tree-walking protocol can be inferred merely by listening to the signals being broadcast by the reader. Therefore, it has been proposed in [75] to encrypt the signals being sent by the reader in order to prevent privacy attacks by eavesdropping of *reader* signals.

A recent approach proposed in [21] makes the observation that the legitimate readers are likely to be much closer to RFID tags, as compared to unauthorized readers which attempt to surreptitiously scan items. It is possible for a tag to detect the strength of the scanning signal, and change its behavior depending upon the distance. For closer readers, the full signal is transmitted, whereas for readers which are further away, only the information about the type of product is transmitted.

A variety of other methods are available to make RFID tags smarter for the purposes of privacy protection. For example, it is possible to modify RFID tags to cycle through a set of *pseudonyms* rather than emit a unique serial number [40]. Thus, the tag cycles through a set of k pseudonyms and emits them sequentially. This makes it more difficult for an attacker to identify the tags, because they may only be able to scan different pseudonyms of the tags at different times. Of course, if the attacker is aware of the method being used in order to mask the tag, they may try to scan the tag over a longer period of time, in order to learn all the pseudonyms associated with the tag. This process can be made more difficult for an attacker by increasing the time it takes for the tag to switch from one pseudonym to another.

Of course, the ability to modify the data in the RFID tags is also a security threat, when it is done by an adversary. Therefore, a natural solution is to password-protect the memory in the RFID tag. This is a challenge from an energy consumption perspective, since all cryptographic algorithms require a large amount of energy, and it would require an onboard battery (active tag) for enablement. In this context,

a number of methods, which have low energy requirements for these cryptographic solutions have been proposed recently [12, 20].

5.5 Privacy Issues in Data Management

In previous subsections, we addressed the privacy issues which arise as a result of eavesdropping on the tag or the reader. In this subsection, we will discuss the privacy issues which arise from the data management issues of the collected data. The general methods for privacy-preservation, such as k-anonymity, ℓ-diversity, t-closeness etc, are also applicable to the data which is captured using RFID technology [3]. The general goal of these methods is to reduce the fidelity of the captured data, so that aggregate inferences can still be derived from it, without compromising privacy.

A number of interesting challenges for privacy arise, when both people and objects are tagged, and the same people have access to the captured RFID data. Such a scenario arises in the context of an RFID Ecosystem constructed at the University of Washington [49, 74]. The most restrictive view to privacy would be one in which users only have access to their own data. While this assures complete privacy of a user, it also unnecessarily curtails the useful insights which one can obtain from such data. This is because events which occurred in the *proximity* of a given user at a given time should be accessible to the user, even if they do not directly relate to the user themselves. This is because such events could be observed by that user by virtue of their physical presence.

It has been observed in [49] that a natural access control policy to use in such a scenario is one in which the data to which a user can gain access is that which corresponds to events which occurred at times and places when and where the user was physically present. This policy is also referred to as *Physical Access Control (PAC)* in [49]. In a sense, such a policy provides a database view which augments people's memory of objects, places and people. It also naturally models the boundaries of people in everyday life. In addition, a user can also specify rules which can relax or restrict the access to data which concerns them. This provides a certain level of personal choice and flexibility in the privacy-preservation process.

The work in [60] further implements the broad principles of the PAC policy by designing a rule-based system, which can infer which information to release for a particular user. The system starts from PAC, and then uses a number of reasoning rules in order to make careful decisions about access control.

6. Conclusions and Summary

While RFID is a relatively old technology, its use for large scale applications has proliferated in recent years. This is because of technological advances in manufacturing, which have made the tags smaller and cheaper. The ability to manufacture a tag at less than 5 cents (per tag) has allowed their widespread use in a cost effective way. RFID data brings numerous challenges with it for the purposes of mining and analysis. RFID data is inherently noisy and redundant because of missed tag readings, or multiple readings of the same tag from different readers. Therefore, techniques need to be designed in order to make the process of reading more robust and reduce the redundancy in the underlying data. The massive volume of the RFID data also makes the process of warehousing and querying the RFID data much more challenging. Therefore methods need to be designed in order to represent the RFID warehouse in terms of the aggregated views of RFID items which typically move together. These aggregated views greatly improve the efficiency of data storage and querying. RFID data can be useful in detecting important semantic events from the underlying data streams. The existing work in active databases and sensor stream event detection can be further extended in a variety of ways to make it suitable to the RFID scenario. For example, methods have recently been designed for event processing in uncertain RFID data streams.

RFID data naturally leads to a number of privacy challenges, because of the association of people with tags, and the likelihood of monitoring people's location with such tags. The privacy issues with RFID data arise *both* during data collection and management. A number of methods such as the kill command, cryptographic protocols, and blocker tags have been designed for privacy protection during data collection. In addition, a number of methods for physical access control have been developed for preserving personal privacy during data management.

References

[1] R. Adaikkalavan, S. Chakravarthy. Snoopib: interval-based event specification and detection for active databases. *Data and Knowledge Engineering*, 59(1): pp. 139–165, 2006.

[2] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[3] C. C. Aggarwal, P. S. Yu. Privacy-Preserving Data Mining,: Models and Algorithms, *Springer*, 2008.

[4] C. C. Aggarwal, T. Abdelzaher. Social Sensing. *Managing and Mining Sensor Data*, Springer, 2013.

[5] C. C. Aggarwal, N. Ashish, A. Sheth. The Internet of Things: A Survey from the Data-Centric Perspective, *Managing and Mining Sensor Data*, Springer, 2013.

[6] J. Agrawal, Y. Diao, D. Gyllstrom, N. Immerman. Efficient Pattern Mining over Event Streams, *ACM SIGMOD Conference*, 2008.

[7] M. Akdere, U. Centintemel, N. Tatbul. Plan-based complex event detection across distributed sources, *VLDB Conference*, 2008.

[8] K. Ashton. That 'Internet of Things' Thing. In: *RFID Journal*, 22 July, 2009.

[9] Y. Bai, F. Wang, P. Liu, C. Zaniolo, S. Liu. RFID Data Stream Processing with a Data Stream Query Language, *ICDE Conference*, 2007.

[10] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, M. Szydylo. Security Analysis of a Cryptographically Enabled RFID Device, *USENIX Security*, 2005.

[11] C. Bornhovd, T. Lin, S. Haller, J. Schaper. Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-Id Infrastructure, *VLDB Conference*, 2004.

[12] B. Calmels, S. canard, M. Girault, H. Sibert. Low-cost cryptography for privacy in RFID systems, *Proceedings of IFIP CARIDS*, 2006.

[13] M. J. Carey, M. Livny, R. Jauhari. The HiPAC project: Combining active databases and timing constraints. In *SIGMOD Record*, 17(1), 1988.

[14] S. Chakravarthy, V. Krishnaprasad, E. Anwar, S. Kim. Composite events for active databases: Semantics, contexts and detection. *VLDB Conference*, pp. 606–617, 1994.

[15] H. Chen, W.-S. Ku, H. Wang, M.-T. Sun. Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing. *ACM SIGMOD Conference*, 2010.

[16] J. Collins. Zebra Unveils RFID Label Maker. *RFID Journal*, September 25, 2003. `http://www.rfidjournal.com/article/articleview/592/1/1`.

[17] N. Dalvi, C. Re, D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1), pp. 25–31, March 2006.

[18] A. Demers, J. Gehrke, M. Hong, M. Riedewald, W.M. White. Towards expressive publish/subscribe systems. *EDBT Conference*, 2006.

[19] N. Dindhar, B. Guc, P. Lau, A. Ozal, M. Soner, N. Tatbul. DejaVu: Declarative Pattern Matching over Live and Archived Streams of Events, *ACM SIGMOD Conference*, 2009.

[20] M. Feldhofer, S. Dominikus, J. Wolkerstorfer. Strong authentication for RFID systems using AES algorithm, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, 2004.

[21] K. Fishkin, S. Roy. Enhancing RFID Privacy via Antenna Energy Analysis, *Tech. Memo IRS–TR–03–012*, Intel Research, Seattle, 2003.

[22] C. Floerkemeier, M. Lampe. RFID Middleware Design – Addressing Application Requirements and and RFID Constraints, *Joint Conference on Smart Objects and Ambient Intelligence*, 2005.

[23] M. J. Franklin, S. R. Jeffrey, S. Krishnamurthy, F. Reiss, E. Wu, O. Cooper, A. Edakkunni, W. Hong. Design Considerations of High Fan-In Systems, *CIDR Conference*, 2005.

[24] S. L. Garfinkel, A. Juels, R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions, *IEEE Security and Privacy*, 3(3), 2005.

[25] M. Garofalakis, K. Brown, M. Franklin, J. Hellerstein, D. Wang, E. Michelakis, L. Tancau, E. Wu, S. Jeffery, R. Aipperspach. Probabilistic Data Management for Pervasive Computing: The Data Furnace Project. *IEEE Data Engineering Bulletin*, 29(1): pp 57–63, 2006.

[26] N. H. Gehani, H. V. Jagadish, O. Shmueli. Composite Event Specification in Active Databases: Model and Implementation, *VLDB Conference*, 1992.

[27] H. Gonzalez, J. Han, X. Shen. Cost-Conscious Cleaning of Massive RFID Data Sets. *ICDE Conference*, 2007.

[28] H. Gonzalez, J. Han, X. Li. Mining compressed commodity workflows from massive RFID data sets. *CIKM Conference*, 2006.

[29] H. Gonzalez, J. Han, X. Li, D. Klabjan. Warehousing and Analyzing Massive RFID Data Sets. *ICDE Conference*, 2006.

[30] H. Gonzalez, J. Han, X. Li. FlowCube: Constructing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows. *VLDB Conference*, 2006.

[31] T. J. Green, V. Tannen. Models for incomplete and probabilistic information. *IEEE Data Engineering Bulletin*, 29(1), pp. 17–24, March 2006.

[32] S. Guiasu, A. Shenitzer. The principle of maximum entropy. *Mathematical Intelligence*, 7(1), 1985.

[33] A. Gupta, M. Srivasatava. Developing auto-id solutions using sun java system rfid software, October 2004.
`http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html`

[34] J. Han, J.-G. Lee, H. Gonzalez, X. Li. Mining Massive RFID, Trajectory, and Traffic Data Sets (Tutorial). *ACM KDD Conference*, 2008.
Video of Tutoral Lecture at: `http://videolectures.net/kdd08_han_mmrfid/`

[35] J. Han, H. Gonzalez, X. Li, D. Klabjan. Warehousing and Mining Massive RFID Data Sets. *ADMA*, 2006.

[36] S. R. Jeffrey, M. Garofalakis, M. J. Franklin. Adaptive Cleaning for RFID Data Streams, *VLDB Conference*, 2006.

[37] S. R. Jeffery, M. J. Franklin, M. Garofalakis. An Adaptive RFID Middleware for Supporting Metaphysical Data Independence. *VLDB Journal*, 17(2), pp. 265–289, 2008.

[38] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. A pipelined framework for online cleaning of sensor data streams. *ICDE Conference*, 2006.

[39] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. Declarative Support for RFID Data Cleaning, *Pervasive*, 2006.

[40] A. Juels. Minimalist Cryptography for RFID Tags. *Conference on Security in Communication Networks*, 2004.

[41] A. Juels, R. Rivest, M. Szydlo. The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy. *ACM Conference on Computer and Communication Security*, pp. 103–111, 2003.

[42] A. Juels, J. Brainard. Soft Blocking: Flexible Blocker Tags on the Cheap, *Workshop on Privacy in the Electronic Society (WPES 04)*, pp. 1–7, 2004.

[43] A. Juels. RFID Security and Privacy: A Research Survey, *IEEE Journal on Selected Areas in Communication*, vol. 24, pp. 381–394, Feb. 2006.

[44] A. Juels, R. Pappu. Squealing Euros: Privacy protection in RFIDenabled banknotes. *Proceedings of Financial Cryptography*, Springer–Verlag, 2003.

[45] N. Khoussainova, M. Balazinska, D. Suciu. Probabilistic Event Extraction from RFID Data, *ICDE Conference*, 2008.

[46] N. Khoussainova, M. Balazinska, D. Suciu. Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. *Fifth*

International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06), 2006.

[47] N. Khoussainova, M. Balazinska, D. Suciu. Peex: Extracting probabilistic events from RFID data. *Technical Report 2007–11–02, Department of Computer Science and Engineering, University of Washington*, 2007.

[48] S. Kinoshita, F. Hoshino, T. Komuro, A. Fujimura, M. Ohkubo. Low-cost RFID privacy protection scheme. *IPS Journal*, 45(8), 2004.

[49] T. Kriplean, E. Welbourne, N. Khoussainova, V. Rastogi, M. Balazinska, G. Borriello, T. Kohno, D. Suciu. Physical Access Control for Captured RFID Data, *IEEE Pervasive Computing*, 6(4), 2007.

[50] J. Landt. The History of RFID. *IEEE Potentials*, October/November 2005.

[51] M. Langheinrich. A Survey of RFID Privacy Approaches. *Personal and Ubiquitous Computing*, Springer, 2008.

[52] G. Liao, J. Li, L. Chen, C. Wen. KLEAP: An Efficient Cleaning Method to Remove Cross-Reads in RFID Data Streams, *ACM CIKM Conference*, 2011.

[53] V. Krishnamurthy, S. Chawathe, S. Ramachandran, S. Sarma. Managing RFID Data, *VLDB Conference*, 2004.

[54] R. McManus. The End of Lost Luggage? RFID Slowly Coming to Airports, *ReadWriteWeb*, 2009.
`http://www.readwriteweb.com/archives/the_end_of_lost_luggage_rfid.php`

[55] D. Molnar, D. Wagner. Privacy and Security in Library RFID: Issues, Practices and Architectures, *CCS*, 2004.

[56] L. Ni, Y. Liu, Y. Lau, A. Patil. LANDMARC: Indoor Location Sensing using Active RFID, *Wireless Networks*, 10(6), pp. 701–710, 2004.

[57] M. Ohkubo, K. Suzuki, S. Kinoshita. RFID Privacy Issues and Technical Challenges, *Communications of the ACM*, 48(9), 2005.

[58] M. Ohkubo, K. Suzuki, S. Kinoshita. A cryptographic approach to "privacy-friendly" tags. *RFID Privacy Workshop*, 2003.

[59] J. Rao, S. Doraiswamy, H. Thakkar, L. S. Colby. A Deferred Cleansing Method for RFID Data Analytics, *VLDB Conference*, 2006.

[60] V. Rastogi, E. Welbourne, N. Khoussainova, T. Kriplean, M. Balazinska, G. Borriello, T. Kohno, D. Suciu. Expressing Privacy Policies using Authorization Views, *International Workshop on Privacy in UbiComp (UbiComp'07)*, 2007.

[61] C. Re, R. Letchner, M. Balazinska, D. Suciu. Event Queries on Correlated Probabilistic Streams, *ACM SIGMOD Conference*, 2008.

[62] S. Rizvi, S. R. Jeffrey, S. Krishnamurthy, M. J. Franklin, N. Burkhart, A. Edakkunni, L. Liang. Events on the edge, *ACM SIGMOD Conference*, 2005.

[63] S. E. Sarma, S. A. Weis, D.W. Engels. Radio-frequency identification systems. *CHES*, pp. 454–469, 2002.

[64] S. E. Sarma, S. A. Weis, D.W. Engels. RFID systems, security and privacy implications. *Technical Report MIT–AUTOID–WH–014*, AutoID Center, MIT, 2002.

[65] L. Sullivan. RFID Implementation Challenges Persist, All This Time Later. *Information Week*, October 2005.

[66] C. Swedberg. Reusable Electronic Baggage Tag Powered by RFID, *RFID Journal*, October 2010.

[67] W. J. Tu, W. Zhou, S. Piramuthu. Identifying RFID-embedded objects in Pervasive Healthcare Applications, *Decision Support Systems*, 46(2), 2009.

[68] F. Wang, P. Liu. Temporal management of RFID Data, *VLDB Conference*, 2005.

[69] F. Wang, S. Liu, P. Liu. Complex RFID Event Processing, *VLDB Journal*, 18(4), 2009.

[70] F. Wang, S. Liu, P. Liu, Y. Bai. Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams, *EDBT Conference*, 2006.

[71] R. Want. An Introduction to RFID Technology, *IEEE Pervasive*, 2006.

[72] R. Want. Enabling Ubiquitous Sensing with RFID, *Computer*, 37(4), pp. 84–86, 2004.

[73] E. Welbourne, M. Balazinska, G. Borriello, W. Brunette. Challenges for Pervasive RFID-based Infrastructures. *IEEE PerCom Workshop on Pervasive RFID/NFC Technology and Applications (PERTEC'07)*, 2007.

[74] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, G. Borriello. Building the Internet of Things Using RFID. *IEEE Internet Computing*, 13(3), May–June, 2009.

[75] S. A. Weis, S. Sarma, R. Rivest, D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. *First International Conference on Security in Pervasive Computing*, 2003.

[76] W. White, M. Riedewald, J. Gehrke, A. Demers. What is "next" in event processing? *ACM PODS Conference*, 2007.

[77] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. *2nd CIDR Conference*, 2005.

[78] E. Wu, Y. Diao, S. Rizvi. High-performance complex event processing over streams. *ACM SIGMOD Conference*, 2006.

[79] D. Zimmer, R. Unland, On the semantics of complex events in active database management systems. *ICDE Conference*, 1999.

[80] iAnyWhere Solutions Inc Whitepaper: *Manage Data Successfully with RFID Anywhere Edge Processing*, `http://www.sybase.com/files/White_Papers/SybaseRFID_edgepro-053107-wp.pdf`.

[81] Hitachi Unveils Smallest RFID Chip. *RFID Journal*, March 14, 2003. `http://www.rfidjournal.com/article/articleview/337/1/1`.

[82] `http://www.boycottbenetton.com`

[83] `http://www.boycotttesco.com`

Chapter 12

THE INTERNET OF THINGS: A SURVEY FROM THE DATA-CENTRIC PERSPECTIVE

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY
charu@us.ibm.com

Naveen Ashish
University of California at Irvine
Irvine, CA
ashish@ics.uci.edu

Amit Sheth
Wright State University
Dayton, OH
amit.sheth@wright.edu

Abstract Advances in sensor data collection technology, such as pervasive and embedded devices, and RFID Technology have lead to a large number of *smart devices* which are connected to the net and continuously transmit their data over time. It has been estimated that the number of internet connected devices has overtaken the number of humans on the planet, since 2008. The collection and processing of such data leads to unprecedented challenges in mining and processing such data. Such data needs to be processed in real-time and the processing may be highly distributed in nature. Even in cases, where the data is stored offline, the size of the data is often so large and distributed, that it requires the use of *big data analytical tools* for processing. In addition, such data is often sensitive, and brings a number of privacy challenges associated

with it. This chapter will discuss a data analytics perspective about mining and managing data associated with this phenomenon, which is now known as the *internet of things*.

Keywords: The Internet of Things, Pervasive Computing, Ubiquitous Computing

1. Introduction

The internet of things [14] refers to uniquely addressable objects and their virtual representations in an Internet-like structure. Such objects may link to information about them, or may transmit real-time sensor data about their state or other useful properties associated with the object. *Radio-Frequency Identification Technology (RFID)* [23, 47, 93, 94] is generally seen as a key enabler of the internet of things, because of its ability to track a large number of uniquely identifiable objects with the use of *Electronic Product Codes (EPC)*. However, other kinds of ubiquitous sensor devices, barcodes, or 2D-codes may also be used to enable the *Internet of Things (IoT)*. The concepts of *pervasive computing* and *ubiquitous computing* are related to the internet of things, in the sense that all of these paradigms are enabled by *large-scale embedded sensor devices*.

The vision of the internet of things is that individual objects of everyday life such as cars, roadways, pacemakers, wirelessly connected pill-shaped cameras in digestive tracks, smart billboards which adjust to the passersby, refrigerators, or even cattle can be equipped with sensors, which can track useful information about these objects. Furthermore, if the objects are *uniquely addressable* and *connected to the internet*, then the information about them can flow through the same protocol that connects our computers to the internet. Since these objects can sense the environment and communicate, they have become tools for understanding complexity, and may often enable autonomic responses to challenging scenarios without human intervention. This broader principle is popularly used in IBM's *Smarter Planet* initiative for autonomic computing.

Since the internet of things is built upon the ability to uniquely identify internet-connected objects, the addressable space must be large enough to accommodate the uniquely assigned IP-addresses to the different devices. The original internet protocol IPv4 uses 32-bit addresses, which allows for only about 4.3 billion unique addresses. This was a reasonable design at the time when IPv4 was proposed, since the total

number of internet connected devices was a small fraction of this number. With an increasing number of devices being connected to the internet, and with each requiring its IP-address (for full peer-to-peer communication and functionality), the available IP-addresses are in short supply. As of 2008, the number of internet connected devices exceeded the total number of people on the planet. Fortunately, the new IPv6 protocol which is being adopted has 128-bit addressability, and therefore has an address space of 2^{128}. This is likely to solve the addressability bottleneck being faced by the internet of things phenomenon.

It is clear that from a data centric perspective, *scalability*, *distributed processing*, and *real time analytics* will be critical for effective enablement. The large number of devices simultaneously producing data in an automated way will greatly dwarf the information which individuals can enter manually. Humans are constrained by time and physical limits in terms of how much a single human can enter into the system manually, and this constraint is unlikely to change very much over time. On the other hand, the physical limitations on how much data can be effectively collected from embedded sensor devices have steadily been increasing with advances in hardware technology. Furthermore, with increasing numbers of devices which are connected to the internet, the number of such streams also continue to increase in time. Simply speaking, automated sensor data is likely to greatly overwhelm the data which are available from more traditional human-centered sources such as social media. In fact, it is the trend towards ubiquitous and pervasive computing, which is the greatest driving force towards *big data analytics*.

Aside from scalability issues, privacy continues to be a challenge for data collection [40, 58–62, 69, 71, 78, 81, 82, 111]. Since the individual objects can be tracked, they can also lead to privacy concerns, when these objects are associated with individuals. A common example in the case of RFID technology is one in which a tagged object (such as clothing) is bought by an individual, and then the individual can be tracked because of the presence of the tag on their person. In cases, where such information is available on the internet, the individual can be tracked from almost anywhere, which could lead to unprecedented violations of privacy.

The material in this chapter is closely related to two other chapters [8, 9] in this book corresponding to *social sensing* and *RFID processing* respectively. However, we have devoted a separate chapter to the internet of things, since it is a somewhat separate concept in its own right, though it is related to the afore-mentioned technologies in the following ways:

- *RFID technology* is a key enabler for the internet of things, because it allows the simultaneous identification of large numbers of objects with cost-effective tags [108]. However, in practice many other kinds of embedded sensor technology may be used for enablement. Furthermore, where more sophisticated sensor information is required about the object, *RFID* technology can only provide a partial component of the data required for full enablement.

- *Social sensing* is a paradigm which refers to the interaction between people with embedded sensor devices, which are typically mobile phones. However, the internet of things is a more general concept, where even mundane objects of everyday life such as refrigerators, consumer products, televisions, or cars may be highly connected, and may be utilized for making smarter and automated decisions.

1.1 The Internet of Things: Broader Vision

The *Internet of Things* is a vision, which is currently being built– there is considerable diversity in its interpretation by different communities, who are involved in an inherently cross-disciplinary effort, involving sensor networking, data management and the world wide web. This diversity is also a result of the technical breadth of the consortiums, industries and communities which support the vision. Correspondingly, this is also reflected in the diversity of the technologies, which are being developed by the different communities. Nevertheless, there are numerous common features across the different visions about what the internet of things may constitute, and it is one of the goals of this paper to bring together these visions from a data-centric perspective.

A simple and broad definition of the internet of things [41, 16] is as follows: "*The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals*". The process of machines communicating with one another, is also referred to as the Machine-to-Machine (M2M) paradigm. This requires tremendous *data-centric* capabilities, which is the primary medium of communication between the different entities. Therefore, the ability to securely and privately collect, manage, index, query and process large amounts of data is critical.

In order to enable these goals, a variety of research efforts have been initiated supporting various aspects of these goals. Each of these visions has a slightly different emphasis on different parts of this data-centric

pipeline. There are three primary visions [16] around which most of the research in this area is focussed:

- **Things-oriented Vision:** This vision is largely supported by the RFID vision of tracking objects with tags [108]. This vision supports the use of the *Electronic Product Code (EPC)* in conjunction with RFID technology to collect and track sensor data. The EPC-global framework [118] is based on this vision of unique product identification and tracking.

  The *things-oriented vision* is by far the dominant vision today, and RFID technology is often (mistakenly) assumed to be synonymous with the internet of things. It is important to note that while RFID technology will continue to be a very important enabler of this phenomenon (especially because of the unique identifiability provided by the EPC), it is certainly not the only technology which can be used for data collection. The things-vision includes data generated by other kinds of embedded sensor devices, actuators, or mobile phones. In fact, more sophisticated sensor technology (beyond tags) is usually required in conjunction with RFID in order to collect and transmit useful information about the objects being tracked. An example of this is the *Wireless Identification and Sensing Platform (WISP)* [121] being constructed at Intel. WISPs are powered by standard RFID readers, and can be used to measure sensing quantities in the physical environment, such as temperature. The overall vision is that of *RFID-based Sensor Networks* [22], which integrate RFID technology, small sensing and computing devices, RFID readers (which provide a key intermediate layer between the "things" and the "internet"), and internet connectivity.

- **Internet-oriented Vision:** The internet-oriented vision corresponds to construction of the IP protocols for enabling *smart objects*, which are internet connected. This is typically spearheaded by the *IPSO* alliance [122]. Typically, this technology goes beyond RFID.

  A theoretical concept, which has emerged in this direction is that of the *spime*, [99] an object, which is uniquely identifiable, and may of its real-time attributes (such as location) can be continuously tracked. Examples of this concept include *smart objects*, which are tiny computers which have sensors or actuators, and a communication device. These can be embedded in cars, light switches, thermometers, billboards, or machinery. Typically these objects

have CPU, memory, a low power communication device, and are battery operated. Since, each of these devices would require its own IP-address, a large part of this vision is also about developing the internet infrastructure to accommodate the ever-expanding number of "things" which require connectivity. A classic example of the efforts in this space include the development of IPv6, which has a much larger addressable IP-space. This vision also supports the development of the *web of things*, in which the focus is to re-use the web-based internet standards and protocols to connect the expanding eco-system of embedded devices built into everyday smart objects [45]. This re-use ensures that widely accepted and understood standards such as URI, HTTP, etc. are used to access the functionality of the smart objects. This approach exposes the synchronous functionality of smart objects through a *REST interface*. The REST interface defines the notion of a resource as any component of an application that is worth being uniquely identified and linked to. On the Web, the identification of resources relies on Uniform Resource Identifiers (URIs), and representations retrieved through resource interactions contain links to other resources [46]. This means that applications can follow links through an interconnected web of resources. Similar to the web, clients of such services can follow these links in order to find resources to interact with. Therefore, a client may explore a service by browsing it, and the services will use different link types to represent different relationships.

- **Semantic-oriented Vision:** The semantic vision addresses the issues of data management which arise in the context of the vast amounts of information which are exchanged by smart objects, and the resources which are available through the web interface. The idea is that standardized resource descriptions are critical to enable interoperability of the heterogeneous resources available through the web of things. The semantic vision is really about the separation of the meanings of data, from the actual data itself. The idea here is that the semantic meanings of objects are stored separately from the data itself, and effective tools for the management of this information. A key capability that this enables in *semantic interoperability and integration* 5semantic i.e., across the sensor data from various sensors.

The diversity of these visions is a result of the diversity in the stakeholders involved in the building of this vision, and also because the vast

infrastructure required by this vision naturally requires the technical expertise from different areas of data analytics, and networking.

This chapter is organized as follows. The next section will discuss applications supported by the internet of things. In section 3, we will present networking issues, and their relationship to the data collection process. Section 4 will discuss issues in data management. This includes methods for querying, indexing, and real-time data analytics. Privacy issues are discussed in section 5. Section 6 contains the conclusions and summary.

2. Applications: Current and Future Potential

The ability of machines and sensors to collect, transmit data and communicate with one another can lead to unprecedented flexibility in terms of the variety of applications which can be supported with this paradigm. While the full potential of the IoT vision is yet to be realized, we will review some of the early potential of existing applications, and also discuss future possibilities. The latter set of possibilities are considered ambitious, but reasonable goals in the longer term, as a part of this broader vision.

Product Inventory Tracking and Logistics This is perhaps one of the most popular applications of the internet of things, and was one of the first large scale applications of RFID technology. The movements of large amounts of products can be tracked by inexpensive RFID tags. For large franchises and organizations, the underlying RFID readers may serve as an intermediate layer between the data collection and internet-connectivity. This provides unprecedented opportunities for product tracking in an automated way. In addition, it is possible to design software, which uses the information from the transmitted data in order to trigger alerts in response to specific events.

Smarter Environment More sophisticated embedded sensor technology can be used in order to monitor and transmit critical environmental parameters such as temperature, humidity, pressure etc. In some cases, RFID technology can be coupled with more sophisticated sensors, in order to send back information which is related to specific objects [106, 107]. Such information can also be used to control the environment in an energy-efficient way. For example, smart sensors in a building can be used in order to decide when the lights or air-conditioning in a room in the building should be switched off, if the room is not currently being used.

Social Sensing Social sensing is an integral paradigm of the internet of things, when the objects being tracked are associated with individual people. Examples of such sensing objects include mobile phones, wearable sensors and piedometers. Such paradigms have tremendous value in enabling social networking paradigms in conjunction with sensing. The increasing ability of commodity hardware to track a wide variety of real-life information such as location, speed, acceleration, sound, video and audio leads to unprecedented opportunity in enabling an increasingly connected and mobile world of users that are ubiquitously connected to the internet. This is also a natural mode in which humans and things can interact with one another in a seamless way over the internet. A detailed discussion on social sensing may be found in [8].

Smarter Devices In the future, it is envisioned that a variety of devices in our day-to-day life such as refrigerators, televisions and cars will be smarter in terms of being equipped with a variety of sensors and will also have internet connectivity in order to publish the collected data. For example, refrigerators may have smart sensors which can detect the quantities of various items and the freshness of perishable items. The internet connectivity may provide the means to communicate with and alert the user to a variety of such information. The user may themselves be connected with the use of one a social sensing device such as a mobile phone. Similarly, sensor equipped and internet connected cars can both provide information to and draw from the repository of data on traffic status and road conditions. In addition, as has recently been demonstrated by the *Google Car* project, sensor-equipped cars have the capability to perform assisted driving for a variety of applications [124]. A further advancement of this technology and vision would be the development of internet connected cars, which can perform automated driving in a way which is sensitive to traffic conditions, with the use of aggregate data from other network connected cars.

Identification and Access Control RFID tags can be used for a wide variety of access control applications. For example, RFID sensors can be used for fast access control on highways, instead of manual toll booths. Similarly, a significant number of library systems have implemented *smart check out* systems with tags on items. When the collected data is allowed to have network connectivity for further (aggregate) analysis and processing, over multiple access points, this also enables significant tracking and analysis capabilities for a variety of applications. For example, in a network of connected libraries, automated tracking can

provide the insights required to decide which books to acquire for the different locations, based on the aggregate analysis.

Electronic Payment Systems Numerous electronic payment systems are now being developed with the use of a variety of smart technologies. The connectivity of RFID readers to the internet can be used in order to implement payment systems. An example is the Texas Instruments's *Speedpass*, pay-at-pump system, which was introduced in Mobil stations in the mid-nineties. This system uses RFID technology in order to detect the identity of the customer buying gas, and this information is used in order to debit the money from the customer's bank account. Another popular payment system, which is becoming available with many mobile phones is based on *Near Field Communications (NFC)*. Many of the latest Android phones have already implemented such systems for mobile payments.

Health Applications RFID and sensor technology have been shown to be very useful in a variety of health applications [100]. For example, RFID chips can be implanted in patients in order to track their medical history. Sensor technology is also very useful in automated monitoring of patients with heart or alzheimer's conditions, assisted living, emergency response, and health monitoring applications [31, 36, 74]. Internet-connected devices can also directly communicate with the required emergency services when required, in order ro respond to emergences, when the sensed data shows the likelihood of significant deterioration in the patient's condition. Smart healthcare technology has the potential to save lives, by significantly improving emergency response times.

3. Networking Issues: Impact on Data Collection

The primary networking issues for the internet of things arise during the data collection phase. At this phase, a variety of technologies are used for data collection, each of which have different tradeoffs in terms of capabilities, energy efficiency, and connectivity, and may also impact both the cleanliness of the data, and how it is transmitted and managed. Therefore, we will first discuss the key networking technologies used for data collection. This will further influence our discussion on data-centric issues of privacy, cleaning and management:

3.1 RFID Technology

At the most basic level, the definition of Radio Frequency Identification (RFID) is as follows: *RFID is a technology which allows a sensor (reader) to read, from a distance, and without line of sight, a unique product identification code (EPC) associated with a tag.* Thus, the unique code from the tag is transmitted to one or more sensor reader(s), which in turn, transmit(s) the readings to one or more server(s). The data at the server is aggregated in order to track all the different product codes which are associated with the tags. We note that such RFID tags do not need to be equipped with a battery, since they are powered by the sensor reader. This is a key advantage from the perspective of providing a high life time to the tracking process. The sensor readers provide a key intermediate layer between the data collection process and network connectivity. The RFID tags typically need to be present at a short distance from the readers in order for the reading process to work effectively. From a data-centric perspective the major limitations of the basic RFID technology are the following:

- The basic RFID technology has limited capabilities in terms of providing more detailed sensing information, especially when passive tags are used.

- The range of the tags is quite small, and is typically of the order of between 5 to 20 meters. As a result significant numbers of readings are dropped.

- The data collected is massively noisy, incomplete and redundant. Sensor readers may repeatedly scan EPC tags which are at the same location (with no addition of knowledge), and multiple readers in the same locality may scan the same EPC tag. This leads to numerous challenges from the perspective of data cleaning. This cleaning typically needs to be performed in the middleware within the sensor reader.

- RFID collection technology leads to considerable privacy challenges, especially when the tags are associated with individual. The tags are susceptible to a wide variety of eavesdropping mechanisms, since covert readers can be used in order to track the locations of individuals.

A detailed discussion of the data-centric issues associated with RFID technology may be found in [9].

3.2 Active and Passive RFID Sensor Networks

The major limitation of the basic RFID sensor technology is that it does not enable detailed sensing information. However, a number of recent methods have been proposed to incorporate sensing into RFID capabilities. One possibility is to use an onboard battery [106, 107] in order to transmit more detailed sensing information about the environment. This is referred to as an *active* RFID tag. Of course, the major limitation of such an approach is that the life-time of the tag is limited by the battery. If a large number of objects are being tracked at given time, then it is not practical to replace the battery or tag on such a basis. Nevertheless, a significant amount of smart object technology is constructed with this approach. The major challenge from the data-centric perspective is to clean or impute the missing data from the underlying collection.

Recently, a number of efforts have focussed on the creating the ability to perform the sensing with *passive* RFID tags. Recently, a number of efforts in this direction [22, 121] are designed to sense more detailed information with the use of passive tags. The major challenge of this approach is that the typical *range* at which the reader must be placed to the tag is even smaller than the basic RFID technology, and may sometimes be less than three meters. This could lead to even more challenges in terms of the dropped readings in a wide variety of application scenarios. On the other hand, since the tag is passive, there are no limitations on the life time because of battery-power consumption.

3.3 Wireless Sensor Networks

A possible solution is to use conventional wireless sensing technology for building the internet of things. One, some, or all nodes in the sensor network may function as gateways to the internet. The major advantage is that peer-to-peer communications among the nodes are possible with this kind of approach. Of course, this kind of approach is significantly more expensive in large-scale applications and is limited by the battery life. The battery-life would be further limited by the fact, that most IP protocols cannot accommodate the sleep modes required by sensor motes in order to conserve battery life. Since the network connectivity of the internet of things is based on the IP protocols, this would require the sensor devices to be on constantly. This would turn out to be a very significant challenge in terms of battery life. The energy requirements can reduced by a variety of methods such as lower sampling or transmission rates, but this can impact the timeliness and quality of the data available for the underlying applications. Wireless sensor networks also

have some quality issues because of the conversion process from voltages to measured values, and other kinds of noise. Nevertheless, from a comparative point of view, wireless sensor networks do have a number of advantages in terms of the quality, range, privacy and security of the data collected and transmitted, and are likely to play a significant role in the internet of things.

3.4 Mobile Connectivity

A significant number of objects in the internet of things, such as mobile phones can be connected by 3G and WiFi connectivity. However, the power usage of such systems is quite high. Such solutions are of course sometimes workable, because such objects fall within the *social sensing* paradigm, where each mobile object belongs to a participant who is responsible for maintaining the battery and other connectivity aspects of the sensing object which is transmitting the data. In such cases, however, the privacy of the transmitted data (eg. GPS location) becomes sensitive, and it is important to design privacy preservation paradigms in order to either limit the data transmission, or reduce the fidelity of the transmitted data. This is of course not desirable from the data analytics perspective, because it reduces the quality of the data analytics output. Correspondingly, the user-trust in the data analytics results are also reduced.

Since mobile phones are usually designed for communication-centric applications, they may only have certain sensors such as GPS, accelerometers, microphones, or video-cameras, which are largely user centric. Also they may allow direct human input into the sensor process. Nevertheless, they do have a number of limitations in not being able to collect arbitrarily kinds of sensed data (eg. humidity). Therefore, the applicability of such devices is often in the context of user-centric applications such as social sensing [8], or working with other smart devices in the context of a broader smart infrastructure.

Since such connectivity has high power requirements, it is important to make the data collection as energy efficient as possible. A salient point to be kept in mind is that data collection can sometimes be performed with the use of multiple methods in the same devices (eg. approximate cell phone tower positioning vs. accurate GPS for location information). Furthermore, tradeoffs are also possible during data *transmission* between timeliness and energy consumption (eg. real-time 3G vs. opportunistic WiFi). A variety of methods have been proposed in recent years, for calibrating these different tradeoffs, so that the energy efficiency is maximized with significantly compromising the data-centric

needs of the application [30, 84, 91, 117]. Examples of specific methods include energy-timeliness tradeoffs [91], adaptive sampling [84], and application-specific collection modes [117]. We note that the impact of such collection policies on data management and processing applications is likely be significant. Therefore, it is critical to design appropriate data cleaning and processing methods, which take such issues of data quality into consideration.

4. Data Management and Analytics

The key to the power of the internet of things paradigm is the ability to provide real time data from many different distributed sources to other machines, smart entities and people for a variety of services. One major challenge is that the underlying data from different resources are extremely heterogeneous, can be very noisy, and are usually very large scale and distributed. Furthermore, it is hard for other entities to use the data effectively, without a clear description of what is available for processing. In order to enable effective use of this very heterogeneous and distributed data, frameworks are required to describe the data in a sufficiently intuitive way, so that it becomes more easily usable i.e., the problem of semantic interoperability is addressed. This leads to unprecedented challenges both in terms of providing high quality, scalable and real time analytics, and also in terms of intuitively describing to users information about what kind of data and services are available in a variety of scenarios. Therefore, methods are required to clean, manage, query and analyze the data in the distributed way. The cleaning is usually performed at data collection time, and is often embedded in the middleware which interfaces with the sensor devices. Therefore, the research on data cleaning is often studied in the context of the *things-oriented vision*. The issues of providing standardized descriptions and access to the data for smart services are generally studied in the context of standardized web protocols and interfaces, and description/querying frameworks such as offered by *semantic web* technology. The idea is to reuse the existing web infrastructure in an intuitive way, so the heterogeneity and distributed nature of the different data sources can be seamlessly integrated with the different services. These issues are usually studied in the context of the *web of things* and the *semantic web* visions. Thus, the end-to-end data management of IoT technology requires the unification and collaboration between the different aspects of how these technologies are developed, in order to provide a seamless and effective infrastructure.

Unlike the world wide web of documents, in which the objects themselves are described in terms of a natural lexicon, the objects and data in the internet of things, are heterogeneous, and may not be naturally available in a sufficiently descriptive way to be searchable, unless an effort is made to create standardized descriptions of these objects in terms of their properties. Frameworks such as RDF provide such a standardized descriptive framework, which greatly eases various functions such as search and querying in the context of the underlying heterogeneity and lack of naturally available descriptions of the objects and the data. Semantic technologies are viewed as a key to resolving the problems of inter-operability and integration within this heterogeneous world of ubiquitously interconnected objects and systems [65]. Thus, the *Internet of Things* will become a *Semantic Web of Things*. It is generally recognized that this interoperability cannot be achieved by making everyone comply to *too many* rigid standards in ubiquitous environments. Therefore, the interoperability can be achieved by designing middleware [65], which acts as a seamless interface for joining heterogeneous components together in a particular IoT application. Such a middleware offers application programming interfaces, communications and other services to applications. Clearly, *some* data-centric standards are still necessary, in order to represent and describe the properties of the data in a homogenous way across heterogeneous environments.

The internet of things requires a plethora of different middlewares, at different parts of the pipeline for data collection and cleaning, service enablement etc. In this section, we will study the data management issues at different stages of this pipeline. First, we will start with data cleaning and pre-processing issues, which need to be performed at data collection time. We will follow this up with issues of data and ontology representation. Finally, we will describe important data-centric applications such as mining with big data analytics, search and indexing.

4.1 Data Cleaning Issues

The data cleaning in IoT technology may be required for a variety of reasons: (a) When is data is collected from conventional sensors, it may be noisy, incomplete, or may require probabilistic uncertain modeling [34]. (b) RFID data is extremely noisy, incomplete and redundant because a large fraction of the readings are dropped, and there are cross-reads from multiple sensor readers. (c) The process of privacy-preservation may require an intentional reduction of data quality, in which case methods are required for privacy-sensitive data processing [6].

Conventional sensor data is noisy because sensor readings are often created by converting other measured quantities (such as voltage) into measured quantities such as the temperature. This process can be very noisy, since the conversion process is not precise. Furthermore, systematic errors are also introduced, because of changes in external conditions or ageing of the sensor. In order to reduce such errors, it is possible to either re-calibrate the sensor [25], or perform data-driven cleaning and uncertainty modeling [34]. Furthermore, the data may sometimes be incomplete because of periodic failure of some of the sensors. A detailed discussion of methods for cleaning conventional sensor data is provided in Chapter 2 of this book.

RFID data is even noisier than conventional sensor data, because of the inherent errors associated with the reader-tag communication process. Furthermore, since RFID data is repeatedly scanned by the reader, even when the data is stationary, it is *massively redundant*. Techniques for cleaning RFID data are discussed in [9]. Therefore, we will provide a brief discussion of these issues and refer the readers to the other chapters for more details. In the context of many different kinds of sources such as conventional sensor data, RFID data, and privacy-preserving data mining, uncertain probabilistic modeling seems to be a solution, which is preferred in a variety of different contexts [6, 34, 66], because of recent advances in the field of probabilistic databases [7]. The broad idea is that when the data can be represented in probabilistic format (which reflects its errors and uncertainty), it can be used more effectively for mining purposes. Nevertheless, probabilistic databases are still an emerging field, and, as far as we are aware, all commercial solutions work with conventional (deterministic) representations of the sensor data. Therefore, more direct solutions are required in order to clean the data as deterministic entities.

In order to address the issue of lost readings in RFID data, many data cleaning systems [47, 120] is to use a temporal smoothing filter, in which a sliding window over the reader's data stream interpolates for lost readings from each tag within the time window. The idea is to provide each tag more opportunities to be read within the smoothing window. Since the window size is a critical parameter, the work in [55] proposes *SMURF (Statistical sMoothing for Unreliable RFid data)*, which is an adaptive smoothing filter for raw RFID data streams. This technique determines the most effective window size automatically, and continuously changes it over the course of the RFID stream. Many of these cleaning methods use *declarative methods* in the cleaning process are discussed in [54, 56, 55]. The broad idea is to specify cleaning stages with the use of high-level declarative queries over relational data streams.

In addition, RFID data exhibits a considerable amount of redundancy because of multiple scans of the same item, even when it is stationary at a given location. In practice, one needs to track only interesting movements and activities on the item. The work in [42] proposes methods for reducing this redundancy. RFID tag readings also exhibit a considerable amount of *spatial redundancy* because of scans of the same object from the RFID readers placed in multiple zones. This is primarily because of the spatial overlap in the range of different sensor readers. This provides seemingly inconsistent readings because of the inconsistent (virtual) locations reported by the different sensors scanning the same object. While the redundancy causes inconsistent readings, it also provides useful information about the location of an object in cases, where the intended reader fails to perform its intended function. The work in [28] proposes a Bayesian inference framework, which takes full advantage of the duplicate readings, and the additional background information in order to maximize the accuracy of RFID data collection.

4.2 Semantic Sensor Web

Sensor networks provide the challenge of too much data, and too little inter-operability and also too little knowledge about the ability to use the different resources which are available in real time. The *Sensor Web Enablement* initiative of the *Open Geospatial Consortium* defines service interfaces which enable an interoperable usage of sensor resources by enabling their discovery, access, tasking, eventing and alerting [21]. Such standardized interfaces are very useful, because such a web hides the heterogeneity of the underlying sensor network from the applications that use it. This initiative defines the term *Sensor Web* as an *"infrastructure enabling access to sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application programming interfaces."* This is critical in order to ensure that the low level sensor details become transparent to application programmers, who may now use higher level abstractions in order to write their applications. Clearly, the goal of the sensor web is to enable real time situation awareness in order to ensure timely responses to a wide variety of events. The main services and language suite specifications include the following:

- *Observations and Measurements (O&M):* These are the standard models and schema, which are used to encode the real-time measurements from a sensor.
- *Sensor Model Language (SML):* These models and schema describe sensor systems and processes. These provide the informa-

tion needed for discovering sensors, locating sensor observations, processing low level sensor observations, and listing taskable properties.

- *Transducer Model Language (TML):* These are standard models and XML schema for describing transducers and supporting real-time streaming of data to and from sensor systems.
- *Sensor Observation Service (SOS):* This is the standard Web service interface for requesting, filtering, and retrieving observations and sensor system information.
- *Sensor Alert Service (SAS):* This is the standard Web service interface for publishing and subscribing to alerts from sensors.
- *Sensor Planning Service (SPS):* This is the standard Web service interface for requesting user-driven acquisitions and observations.
- *Web Notification Services (WNS):* This is the standard Web service interface for delivery of messages or alerts from *Sensor Alert Service* and *Sensor Planing Services*.

We note that all of the above services are useful for different aspects of sensor data processing, and this may be done in different ways based on the underlying scenario. For example, the *discovery* of the appropriate sensors is a critical task for the user, though it is not always easy to know a-priori about the nature of the discovery that a user may request. For example, a user may be interested in discovering physical sensors based on specific criteria such as location, measurement type, semantic meta-information etc., or they may be interested in specific sensor related functionality such as alerting [57]. Either goal may be achieved with an appropriate implementation of the SML module [21, 57]. Thus, the specific design of each module will dictate the functionality which is available in a given infrastructure.

The World Wide Web Consortium (W3C) has also initiated the *Semantic Sensor Networks Incubator Group (SSN-XL)* to develop Semantic Sensor Network Ontologies, which can model sensor devices, processes, systems and observations. This ontology enables expressive representation of sensors, sensor observations, and knowledge of the environment. This is already being adopted widely by the sensor networking community, and has resulted in improved management of sensor data on the Web, involving annotation, integration, publishing, and search. In the case of sensor data, the amounts of data are so large, that the semantic annotation of the underlying data is extremely important in order to enable effective discovery and search of the underlying resources. This

annotation can be either spatial, temporal, or may be semantic in nature. Interesting discussions of research issues which arise in the context of the semantic and database management issues of the sensor web may be found in [96, 17].

The semantic web encodes meta-data about the data collected by sensors, in order to make it effectively searchable and usable by the underlying services. This comprises the following primary components:

- The data is encoded with self-describing XML identifiers. This also enables a standard XML parser to parse the data.
- The identifiers are expressed using the *Resource Description Framework (RDF).* RDF encodes the meaning in sets of triples, with each triple being a subject, verb, and object of an element. Each element defines a Uniform Resource Identifier on the Web.
- Ontologies can express relationships between identifiers. For example, one accelerometer sensor, can express the speed in miles per hour, whereas another will express the speed in terms of Kilometers per hour. The ontologies can represent the relationships among these sensors in order to be able to make the appropriate conversion.

We will describe each of these components in the description below.

While the availability of real-time sensor data on a large scale in domains ranging from traffic monitoring to weather forecasting to homeland security to entertainment to disaster response is a reality today, major benefits of such sensor data can only be realized if and only if we have the infrastructure and mechanisms to synthesize, interpret, and apply this data intelligently via automated means. The Semantic Web vision [73] was to make the World Wide Web more intelligent by layering the networked Web content with semantics. The idea was that a semantic layer would enable the realization of automated agents and applications that "understand" or "comprehend" Web content for specific tasks and applications. Similarly the Semantic Sensor Web puts the layer of intelligence and semantics on top of the deluge of data coming from sensors. In simple terms, it is the Semantic Sensor Web that allows automated applications to understand, interpret and reason with basic but critical semantic notions such as "nearby", "far", "soon", "immediately", "dangerously high", "safe", "blocked", or "smooth", when talking about data coming from sensors, and the associated geo-spatial and spatio-temporal reasoning that must accompany it. In summary, it enables true semantic interoperability and integration over sensor data. In this section, we describe multiple aspects of Semantic Sensor Web

technology that enables the advancement of sensor data mining applications in a variety of critical domains.

4.2.1 Ontologies. Ontologies are at the heart of any semantic technology, including the Semantic Sensor Web. An ontology, defined formally as a specification of a conceptualization [43], is a mechanism for knowledge sharing and reuse. In this chapter, we will illustrate two important ontologies that are particularly relevant to the sensor data domain. Our aim is to provide an understanding of ontologies and ontological frameworks per se, as well as highlight the utility of existing ontologies for (further) developing practical sensor data applications. Ontologies are essentially knowledge representation systems. Any knowledge representation system must have mechanisms for (i) Representation and (ii) Inference. In this context, we provide a brief introduction to two important Semantic-Web ontology representation formalisms - namely *RDF* and *OWL*.

RDF stands for the "*Resource Description Framework*" and is a language to describe resources [76]. A resource is literally any thing or concept in the world. For instance, it could be a person, a place, a restaurant entree etc. Each resource is uniquely identified by a *URI*, which corresponds to a Unique Resource Identifier. What RDF enables us to do is to:

- Unambiguously describe a concept or a resource.
- Specify how resources are related.
- Do inferencing.

The building blocks of RDF are *triples*, where a triple is a 3-tuple of the form $< subject, predicate, object >$ where subject, predicate and object are interpreted as in a natural language sentence. For instance the triple representation of the sentence "*Washington DC is the capital of the United States*" is illustrated in Figure 12.1.

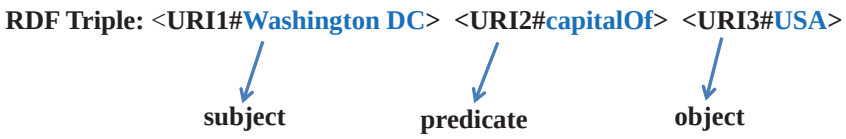


Figure 12.1. RDF Triples

The subject and predicate must be resources. This means that they are things or concepts having a URI. The object however can be a resource or a literal (such as the string "USA" or the number "10").

It is most helpful to perceive RDF as a graph, where subject resources are represented in ovals, literals in rectangles, and predicate (relationships) represented as directed edges between ovals or between ovals and rectangles. An example is illustrated in Figure 12.2.

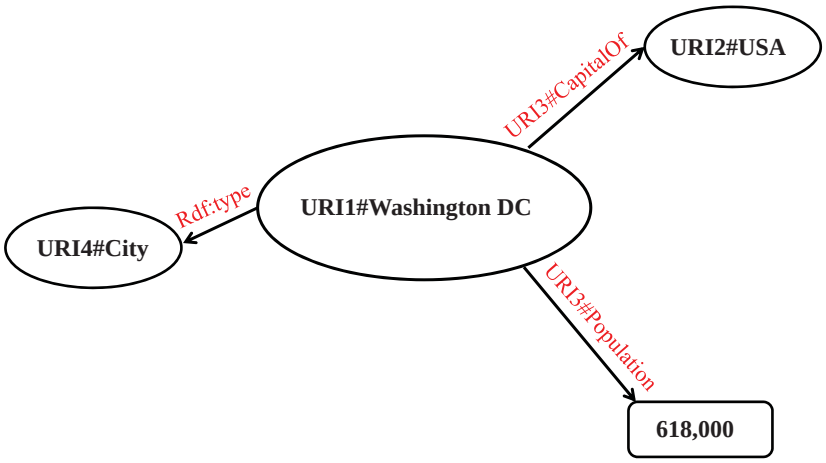


Figure 12.2. RDF as a Graph

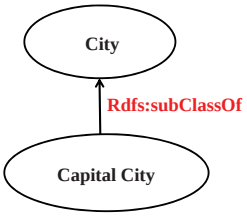
The most popular representation for RDF is RDF/XML. In this case, the RDF is represented in XML format, as illustrated in Figure 12.3, where XML elements are used to capture the fundamental resources and relationships in any RDF triple.

```
<rdf:Descriptionrdf:about="URI1#WashingtonDC">
<rdf:typerdf:resource="URI4#City"/>
<URI2#isCapitalOfrdf:resource="URI3#USA"/>
</rdf:Description>
```

Figure 12.3. RDF XML Representation

RDF(S) stands for RDF *(Schema)* [76]. This can be viewed as a metamodel that is used to define the vocabulary used in an RDF document.

RDF(S) is used for defining classes, properties, hierarchies, collections, reification, documentation and basic entailments for reasoning.




```
<rdfs:Classrdf:ID="someURI#CapitalCity">
        <rdfs:subClassOfrdf:resource="someURI#City"/>
</rdfs:Class>
```

Figure 12.4. RDF Schema

For instance, let us say that we need to define a separate collection of cities that are capital cities of any country. A capital city is of course a sub-class of cities in general. This is represented in RDF(S) as shown in Figure 12.4.

OWL stands for *Web Ontology Language* [76]. This is another ontology formalism that was developed to overcome the challenges with RDF. RDF (and RDF Schema) are limited in that they do not provide ways to represent constraints (such as domain or range constraints). Further, transitive, inverse or closure properties cannot be represented in RDF(S). Extending RDF(s) with the use of standards (XML, RDF etc.,), making it easy to use and understand, and providing a Formal specification is what results in OWL. Both RDF and OWL ontology formats have extensive developer community support in terms of the availability of tools for ontology creation and authoring. An example is *Protege* [101], which supports RDF and OWL formats, data storage and management stores such as *OpenSesame*, for efficient storage and querying of data in RDF or OWL formats. Furthermore, there is significant availability of actual ontologies in a variety of domains in the RDF and OWL formats.

Specific ontologies: We now describe two such ontologies – *SSN* [119] and *SWEET* [92] that are particularly relevant to sensor data semantics. Both these ontologies have been created with the intention of being generic and widely applicable for practical application tasks. SSN is more sensor management centric, whereas SWEET has a particular

focus on earth and environmental data (a vast majority of the data collected by sensors). The *Semantic Sensor Network (SSN)* ontology [119] is an OWL ontology developed by the W3C Semantic Sensor Network Incubator group (the SSN-XG) [119] to describe sensors and observations. The SSN ontology can describe sensors in terms of their capabilities, measurement processes, observations and deployments. The SSN ontology development working group (SSN-XG) targeted the SSN ontology development towards four use cases, namely (i) Data discovery and linking, (ii) Device discovery and selection, (iii) Provenance and diagnosis, and (iv) Device operation, tasking and programming. The SSN ontology is aligned with the DOLCE Ultra Lite (DUL) *upper ontology* [39] (an upper ontology is an ontology of more generic, higher level concepts that more specific ontologies can anchor their concepts to) . This has helped to normalize the structure of the ontology to assist its use in conjunction with ontologies or linked data resources developed elsewhere. DUL was chosen as the upper ontology because it is more lightweight than other options, while having an ontological framework and basis. In this case, qualities, regions and object categories are consistent with the group's modeling of SSN. The SSN ontology itself, is organized, conceptually but not physically, into ten modules as shown in Figure 12.5. The SSN ontology is built around a central Ontology Design Pattern (ODP) describing the relationships between sensors, stimulus, and observations, the Stimulus-Sensor- Observation (SSO) pattern. The ontology can be seen from four main perspectives:

- A sensor perspective, with a focus on what senses, how it senses, and what is sensed.
- An observation perspective, with a focus on observation data and related metadata.
- A system perspective, with a focus on systems of sensors and deployments.
- A feature and property perspective, focusing on what senses a particular property or what observations have been made about a property.

The full ontology consists of 41 concepts and 39 object properties, directly inherited from 11 DUL concepts and 14 DUL object properties. The ontology can describe sensors, the accuracy and capabilities of such sensors, observations and methods used for sensing. Concepts for operating and survival ranges are also included, as these are often part of a given specification for a sensor, along with its performance within

those ranges. Finally, a structure for field deployments is included to describe deployment lifetimes and sensing purposes of the deployed macro instrument.

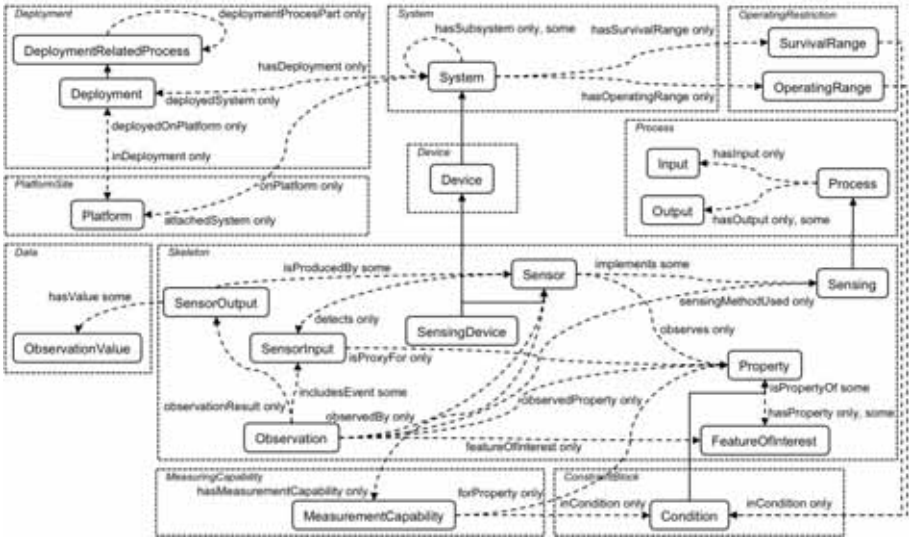


Figure 12.5. The Ten Modules in the SSN Ontology

```
<owl:Classrdf:about="http://purl.oclc.org/NET/ssnx/ssn#SensingDevice">
<rdfs:label>SensingDevice</rdfs:label>
<rdfs:subClassOfrdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Device"/>
<rdfs:subClassOfrdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Sensor"/>
<dc:source>http://www.w3.org/2005/Incubator/ssn/</dc:source>
<rdfs:comment>A sensing device is a device that implements sensing.</rdfs:comment>
<rdfs:isDefinedBy>http://purl.oclc.org/NET/ssnx/ssn</rdfs:isDefinedBy>
<rdfs:seeAlso>
http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Sensor#Measuring
</rdfs:seeAlso>
</owl:Class>
```

Figure 12.6. Schema for the Sensor Class

SWEET: The motivation for developing *SWEET* (*The Semantic Web of Earth and Environmental Terminology*) stemmed from the realization of making vast amounts of earth science related sensor data collected continuously by NASA more understandable and useful [92]. This effort

resulted in a) a collection of ontologies for describing Earth science data and knowledge, and b) an ontology-aided search tool to demonstrate the use of these ontologies. The set of keywords in the NASA Global Change Master Directory (GCMD) (Global Change Master Directory, 2003) form the starting point for the SWEET ontology. This collection includes both controlled and uncontrolled keywords. The controlled keywords include approximately 1000 Earth science terms represented in a subject taxonomy. Several hundred additional controlled keywords are defined for ancillary support, such as: instruments, data centers, missions, etc. The controlled keywords are represented as a taxonomy. The uncontrolled keywords consist of 20,000 terms submitted by data providers. These terms tend to be more general than or synonymous with the controlled terms. Examples of frequently submitted terms include: climatology, remote sensing, EOSDIS, statistics, marine, geology, vegetation, etc.

Some of the SWEET ontologies represent the Earth realm and phenomena and/or physical aspects and phenomena. These include the "Earth Realm" ontology which has elements related to "atmosphere", "ocean" etc., Physical aspects ontologies represent things like substances, living elements and physical properties. However the ontologies most relevant to sensor data are those representing (i) Units, (ii) Numerical entities, (iii) Temporal entities, (iv) Spatial entities, and (v) Phenomena.

4.2.2 Query Languages. While RDF, OWL and other formalisms serve the purpose of data and knowledge representation, one also needs a mechanism for querying any data and knowledge stored. SPARQL (SPARQL Protocol and RDF Query Language) [88] is an RDF query language for querying and manipulating data stored in the RDF format. SPARQL allows writing queries over data as perceived as triples. It allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns. SPARQL closely follows SQL syntax. As a result, its query processing mechanisms are able to inherit from standard database query processing techniques. A simple example of an SPARQL query, which returns the name and email of every person in a data set is provided in Figure 12.7. Significantly, this query can be distributed to multiple SPARQL endpoints for computation, gathering and generation of results. This is referred to as a *Federated Query*.

SPARQLstream [89] is an extension of SPARQL that facilitates querying over RDF streams. This is particularly valuable in the context of *sensor data*, which is generally stream-based. An RDF stream is defined as a sequence of pairs (T_i, i) where T_i is an RDF triple $< h_{si}; p_i; o_{ii} >$ and i is a time-stamp which comes from a monotonically non-decreasing sequence.

```
PREFIXfoaf:<http://xmlns.com/foaf/0.1/>
SELECT?name?email
WHERE{
?personafoaf:Person.
?personfoaf:name?name.
?personfoaf:mbox?email.
}
```

Figure 12.7. Simple SPARQL Example

An RDF stream is identified by an IRI, which provides the location of the data source. An example SPARQL stream query is provided in Figure 12.8 whichillustrates a query that obtains all wind-speed observation values greater than some threshold (e.g., 10) in the last 5 hours, from the *sensors* virtual rdf stream **swissex:WannengratWindSensors.srdf**.

```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX swissex:<http://swis-experiment.ch/metadata#>
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#>
PREFIX sweetSpeed:<http://sweet.jpl.nasa.gov/2.1/propSpeed.owl#>
SELECT ?speed ?obs
FROM NAMED STREAM swissex:WannengratWindSpeed.srdf[NOW--5HOUR]
WHERE {
?obs      assn:Observation;
          ssn:observationResult?result;
          ssn:observedProperty?prop.
?prop     asweetSpeed:WindSpeed.
?result   ssn:hasValue?obsvalue.
?obsvalue          assn:ObservationValue;
          qudt:numericValue?
FILTER ( ? speed > 10 ) }
```

Figure 12.8. SPARQL Stream Example

4.2.3 Linked Data. Realization of the Semantic-Web vision has indeed faced challenges on multiple fronts, some impediments including having to define and develop ontologies that domain experts and representatives can agree upon, ensuring that data on the Web is indeed marked up in semantic formats, etc. The *Linked Data* vision

[109] is a more recent initiative that can perhaps be described as a "light-weight" Semantic Web. In a nutshell, Linked Data describes a paradigm shift from a Web of linked documents towards a Web of linked data. Flexible, minimalistic, and local vocabularies are required to interlink single, context-specific data fragments on the Web. In conjunction with ontologies, such raw data can be combined and reused on-the-fly. In comparison to SDIs, the Linked Data paradigm is relatively simple and, therefore, can help to open up SDIs to casual users. Within the last years, Linked Data has become the most promising vision for the Future Internet and has been widely adopted by academia and industry. The Linking Open Data cloud diagram provides a good and up-to-date overview. Some of the foundational work for taking sensor data to the Linked Data paradigm has been in the context of Digital Earth [109], which calls for more dynamic information systems, new sources of information, and stronger capabilities for their integration. Sensor networks have been identified as a major information source for the Digital Earth, while Semantic Web technologies have been proposed to facilitate integration. So far, sensor data is stored and published using the Observations and Measurements standard of the Open Geospatial Consortium (OGC) as data model. With the advent of Volunteered Geographic Information and the Semantic Sensor Web, work on an ontological model gained importance within Sensor Web Enablement. In contrast to data models, an ontological approach abstracts from implementation details by focusing on modeling the physical world from the perspective of a particular domain. Ontologies restrict the interpretation of vocabularies towards their intended meaning. The ongoing paradigm shift towards Linked Sensor Data complements this attempt. Two questions need to be addressed:

- How to refer to changing and frequently updated data sets using Uniform Resource Identifiers.

- How to establish meaningful links between those data sets, i.e., observations, sensors, features of interest, and observed properties?

The work in [109] presents a Linked Data model and a RESTful proxy for OGC's Sensor Observation Service to improve integration and interlinkage of observation data for the Digital Earth.

In summary, today with the existence of practical and real-world sensor domain ontologies (such as SSN and SWEET), RDF storage and streaming query language mechanisms, and the availability of linked sensor data - we are in a position to use such infrastructure for building practical sensor data mining applications.

4.3 Semantic Web Data Management

One of the most challenging aspects of RDF data management is that they are represented in the form of *triples* which conceptually represents a graph structure of a particular type. The conventional method to represent RDF data is in the form of triple stores. In these cases, giant triples tables are used in order to represent the underlying RDF data [11, 12, 18, 24, 49, 50, 85, 113, 114]. In these systems, the RDF data is decomposed into a large number of statements or triples that are stored in conventional relational tables, or hash tables. Such systems can effectively support statement-based queries, in which the query is missing some parts of the triple, and these parts are then provided by the response. On the other hand, many queries cannot be answered from a single property table, but from multiple property tables. One major problem with such solutions is that because a relational structure is imposed on inherently structured data, it results in sparse tables with many null values. This causes numerous scalability challenges, because of the computational overhead in processing such sparse tables.

A natural solution is to index the RDF data directly as a graph. This has the virtue of recognizing the inherently structured nature of the data for storage and processing [13, 20, 53, 103]. A number of graph-based methods also use the measurement of similarity within the Semantic Web [67], and selectivity estimation techniques for query optimization of RDF data [97]. Many of these techniques require combinatorial graph exploration techniques with main memory operations necessitated by the random storage access inherent in graph analytics. Such approaches can doom the scalability of RDF management. Other methods use path-based techniques [68, 75] for storing and retrieving RDF data. These methods essentially store subgraphs into relational tables. As discussed earlier, approaches which are based on relational data have fundamental limitations which cannot be addressed by these methods.

A different approach is to use multiple indexing approaches [51, 116] in which information about the context is added to the triple. Thus, we now have a *quad* instead of a *triple* which has $2^4 = 16$ possible access patterns. The work in [51] creates six indexes which cover all these 16 access patterns. Thus, a query, which contains any subset of these variables can be easily satisfied with this approach. These methods are also designed for statement-based queries, and do not provide efficient support for more complex queries.

4.3.1 Vertical Partitioning Approach. A fundamental paradigm shift in the management of RDF data is with the use of a

vertical partitioning approach [1]. This is closely related to the development of column-oriented databases for sensor data management [98, 2, 3]. Consider a situation in which we have m different properties in the data. In such a case, a total of m two-column tables are created. Each table contains a subject and object column, and if a subject is related to multiple objects, this corresponds to the different rows in the table. The tables may be stored by subject, and this can enable quick location of a specific subject. Furthermore, each table is sorted by subject, so that particular subjects can be located quickly, and fast merge-joins can be used to reconstruct information about multiple properties for subsets of subjects. This approach is combined with a column-oriented database system [98] in order to achieve better compression and performance. In addition, the object columns of the scheme can be indexed with the use of a B^+-Tree or any other index. It was argued in [110] that the scheme in [1] is also not particularly effective, unless the properties appear as bound variables.

It was observed in [110] that while the work in [1] argued against conventional property-table solutions, their solution turned out to be a special variation of property tables, and therefore share all its disadvantages. The two-column tables of [1] are similar to the multi-valued property tables introduced in [113], and the real novelty of the work in [1] was to integrate the column-oriented database systems into two-column property tables. Therefore, the work in [110] combines a multiple-indexing scheme with the vertical partitioning approach proposed in [1] in order to obtain more effective results. The use of multiple indexes has tremendous potential to be extremely effective for semantic web management, because of its simultaneous exploitations of different access patterns, while incorporating the virtues of a vertical approach. Multiple index-based techniques have also been used successfully for a variety of other database applications such as join processing [15, 79, 80].

4.4 Real-time and Big Data Analytics for The Internet of Things

Since RFID and conventional sensors form the backbone of the data collection mechanisms in the internet of things, the volume of the data collected is likely to be extremely large. We note that this large size is not just because of the streaming nature of the collected data, but also because smart infrastructures typically have a large number of objects simultaneously collecting data and communicating with one another. In many cases, the communications and data transfers between the objects may be required to enable smart analytics. Such communications and

transfers may require both bandwidth and energy consumption, which are usually a limited resource in real scenarios. Furthermore, the analytics required for such applications is often real-time, and therefore it requires the design of methods which can provide real-time insights in a distributed way, with communication requirements. Discussions of such techniques for a wide variety of data mining problems can be found in the earlier chapters of thus book, and also in [5].

In addition to the real-time insights, it is desirable to glean *historical* insights from the underlying data. In such cases, the insights may need to be gleaned from massive amounts of archived sensor data. In this context, Google's *MapReduce* framework [33] provides an effective method for analysis of the sensor data, especially when the nature of the computations involve linearly computable statistical functions over the elements of the data streams (such as MIN, MAX, SUM, MEAN etc.). A primer on the *MapReduce* framework implementation on *Apache Hadoop* may be found in [115]. Google's original *MapReduce* framework was designed for analyzing large amounts of web logs, and more specifically deriving such linearly computable statistics from the logs. Sensor data has a number of conceptual similarities to logs, in that they are similarly repetitive, and the typical statistical computations which are often performed on sensor data for many applications are linear in nature. Therefore, it is quite natural to use this framework for sensor data analytics.

In order to understand this framework, let us consider the case, when we are trying to determine the maximum temperature in each year, from sensor data recorded over a long period of time. The *Map* and *Reduce* functions of *MapReduce* are defined with respect to data structured in $(key, value)$ pairs. The *Map* function, takes a list of pairs (k_1, v_1) from one domain, returns a list of pairs (k_2, v_2). This computation is typically performed in parallel by dividing the key value pairs across different distributed computers. For example, in our example above consider the case, where the data is in the form of $(year, value)$, where the year is the key. Then, the *Map* function, also returns a list of $(year, local\_max\_value)$ pairs, where $local\_max\_value$ represents the local maximum in the subset of the data processed by that node.

At this point, the *MapReduce* framework collects all pairs with the same key from all lists and groups them together, thus creating one group for each one of the different generated keys. We note that this step requires communication between the different nodes, but the cost of this communication is much lower than moving the *original* data around, because the Map step has already created a compact summary from the data processed within its node. We note that the exact implementation

of this step depends upon the particular implementation of *MapReduce* which is used, and exact nature of the distributed data. For example, the data may be distributed over a local cluster of computers (with the use of an implementation such as *Hadoop*), or it may be geographically distributed because the data was originally created at that location, and it is too expensive to move the data around. The latter scenario is much more likely in the IoT framework. Nevertheless, the steps for collecting the intermediate results from the different *Map* steps may depend upon the specific implementation and scenario in which the *MapReduce* framework is used.

The *Reduce* function is then applied in parallel to each group, which in turn produces a collection of values in the same domain. Next, we apply $Reduce(k2, list(v2))$ in order to create $list(v3)$. Typically the Reduce calls over the different keys are distributed over the different nodes, and each such call will return one value, though it is possible for the call to return more than one value. In the previous example, the input to *Reduce* will be a list of the form $(Year, [local\_max1, local\_max2, \ldots local\_maxr])$, where the local maximum values are determined by the execution of the different *Map* functions. The *Reduce* function will then determine the maximum value over the corresponding list in each call of the *Reduce* function.

The *MapReduce* framework is very powerful in terms of enabling distributed search and indexing capabilities across the semantic web. An overview paper in this direction [77] explores the various data processing capabilities of *MapReduce* used by $Yahoo!$ for enabling efficient search and indexing. The *MapReduce* framework has also been used for distributed reasoning across the semantic web [104, 105]. The work in [105] addresses the issue of semantic web compression with the use of the *MapReduce* framework. The work is based on the fact that since the number of RDF statements are rapidly increasing over time (because of a corresponding increase in the number of "things"), the compression of these strings would be useful for storage and retrieval. One of the most often used techniques for compressing data is called *dictionary encoding*. It has been experimentally estimated that the statements on the semantic web require about 150–210 bytes. If this text is replaced with 8 byte numbers, the same statement requires only 24 bytes, which is a significant saving. The work in [105] presents methods for performing this compression with the use of the *MapReduce* framework. Methods for computing the closure of the RDF graph with the use of the *MapReduce* framework are proposed in [104].

The *Hadoop* implementation of the *MapReduce* framework is an open source implementation provided by *Apache*. This framework implements

a *Hadoop Distributed File System (HDFS)*, which is similar to Google's file system. HDFS provides a distributed file system, in which data is distributed across multiple machines, with some replication, in order to provide resilience to disk failures. The Hadoop framework handles the process of task sub-division, and mapping the *Map* and *Reduce* subtasks to the different machines. This process is completely transparent to the programmer, who can focus their attention on building the *Map* and *Reduce* functions. There are two other related big-data technologies which are very useful for data management in the semantic web.

HBase The *HBase* is a database abstraction within the *Hadoop* framework, which is similar to the original *BigTable* system [27, 126]. The *HBase* has column which serves as the key, and is the only index which may be used to retrieve the rows. The data in *HBase* is also stored as $(key, value)$ pairs, where the content in the non-key columns may be considered the values.

Pig The *Pig* implementation builds upon the *Hadoop* framework in order to provide further database-like functionality. A table in *Pig* is a set of tuples, and each field is either a value or a set of tuples. Thus, this framework allows for nested tables, which is a rather powerful abstraction. *Pig* also provides a scripting language [83] called *PigLatin*, which provides all the familiar constructs of SQL such as projections, joins, sorting, grouping etc. Different from SQL, *PigLatin* scripts are *procedural*, and are rather easy for programmers to pick up. The *PigLatin* language provides a higher abstraction level to the *MapReduce* framework, because a query in *PigLatin* can be transformed into a sequence of *MapReduce* jobs.

One interesting aspect of *Pig* is that its data model and transformation language are similar to RDF and the SPARQL query language respectively. Therefore, *Pig* was recently extended [77] to perform RDF querying and transformations. Specifically, *Load* and *Save* functions were defined to convert RDF into Pig's data model, and a complete mapping was created between SPARQL and *PigLatin*.

All of these technologies play a very useful role in crawling storing and analyzing the massive RDF data sets, which are possible and likely in the massive scale involved in the internet of things. In the next subsection, we will discuss some of the ways in which these technologies can be used for search and indexing.

4.5 Crawling and Searching the Internet of Things

The Internet of Things is the beginning of the *data-centric web era*, where the data could be about events, locations or people, as is collected by the sensor infrastructure, and richly described in the form of RDF meta-data. Therefore, it is natural to move to the next stage of *smart semantic web search*, where data and services about arbitrary "things" such as people, events and locations can be easily accessed. Providing such search functionality will be *extremely* challenging, because the size of the semantic web continues to grow rapidly, and is expected to be several *orders of magnitude* larger than the conventional web. This leads to numerous challenging in crawling, indexing and retrieving search results on the semantic web. While the RDF framework solves the *representation issues* for effective search and indexing, the data scalability issue continues to be an enormous challenge. Nevertheless, such a functionality is critical, because search engines can locate the data and services that other applications may need in a M2M world.

Some early frameworks for semantic web search may be found in [44, 72]. Some real implementations of meta-data search engines are *Swoogle* [35, 129] and *Sindice* [102, 127]. Among these different frameworks and implementations, only the last one is recent enough to incorporate the full advantages of the *MapReduce* framework. Generally speaking, since the semantic web is similar to the conventional web in terms of being a linked entity, algorithms which are similar to *PageRank* can be implemented with a *MapReduce* framework for efficient retrieval. The semantic web may require slightly more sophisticated algorithms for indexing, as compared to the conventional web, because of the greater richness in the semantic web in terms of accommodating different types of links. Other tasks such as crawling, are also very similar to the conventional web, in terms of using the linkage structure during the crawling process. Again, some additional intelligence may be incorporated into the crawling process, depending upon the importance of different links and crawling strategies for resource discovery.

A very recent large-scale framework for search and indexing of the web is *Sindice* [102, 127]. We will discuss this engine in more detail, because the high level of scalability, which is incorporated in all aspects of its design choices. In particular, this is achieved with the use of the *MapReduce* framework. The first step is to harvest the web with a crawler called *SindiceBot*, that collects web and RDF documents. This crawler utilizes *Hadoop* in order to distribute the crawling job across multiple machines. An extension to the *Sitemap* protocol [128] allows the data sets to be

a described in such a way, that they can be downloaded as a dump, rather than having to download each references URI individually. Nevertheless, the processing of such dumps in order to create indexed RDF representations is computationally intensive. This is achieved with the use of the *MapReduce* framework [127].

In order to create the index, the first step is to process the raw data from *HBase*. The semantics of the raw data are extracted and represented in RDF. At this point, reasoning is applied to fact sets in order to increase the richness of the indexing for query processing purposes. Finally entities are consolidated with appropriate cross-references between the data and its index.

Once the index is created, traditional information retrieval techniques are used in order answer textual and semantic queries over large collections of documents. We note that this phase is relatively efficient, once the index has been materialized, and does not necessarily require the use of the *MapReduce* framework. However, the initial stage of crawling, processing and indexing the data is extremely computationally intensive, and cannot be easily achieved without efficient distributed techniques.

5. Privacy and Security

Privacy and security are an important concern in systems, which are as open as the internet of things. The issues of data privacy may arise both during data collection, and during data transmission and sharing. Privacy in data *collection* issues typically arise because of the widespread use of RFID technology, in which the tags carried by a person may become a unique identifier for that person. Privacy in data *sharing and management* may arise because much of the information being transmitted (eg. GPS location) can be sensitive, but it may also be required (on an *aggregate* basis) to enable useful real-time applications such as traffic analysis. In this section, we will discuss both issues. In addition, a number of security issues also arise involving the access control of the managed data. We will discuss these issues below.

5.1 Privacy in Data Collection

As discussed above, the ability to track the RFID data with covert readers is a significant challenge in the data collection process. We have discussed details of methods for reducing the privacy risks in the data collection process in the chapter on RFID processing in this book [9]. In this section, we will provide an abbreviated discussion about these issues. Once an RFID-based smart object is carried by a user on their person (as would be natural in many applications), the EPC then

becomes a unique identifier for that person. The information about object movement can be used either to track the whereabouts of the person, or even for corporate espionage in a product supply chain.

The simplest solution to privacy with RFID data is the use of the *kill* command. The Auto-Id Center designed the "kill" command, which are intended to be executed at the point of sale. The kill command can be triggered by a signal, which explicitly disables the tag [63, 64]. If desired, a short 8-bit password can be included with the "kill" command. The tag is subsequently "dead" and no longer emits the EPC, which is needed to identify it. However, the killing of a tag, was mostly designed for cases where tags were associated with products, which have a limited lifespan (before point of sale) for tracking purposes. This may not work with smart products, where the tags are essential to its functioning over the entire lifetime [40]. Another mechanism is to use a locking and unlocking mechanism for the tags [111], if the data collection from the tag is known to be needed only in specific periods, where the data collection is relatively secure from eavesdropping. This can work in some smart applications, where such periods are known in advance.

More robust solutions are possible with cryptographic methods. For example, it is possible to encrypt the code in a tag before transmission. However, such a solution may not be very effective, because this only protects the *content* of the tag, but not the ability to *uniquely identify* the tag. For example, the encoded tag is itself a kind of meta-tag, which can be used for the purposes of tracking. Another solution is to embed dynamic encryption ability within the tag. Such a solution, however, comes at a cost, because it requires the chip to have the ability to perform such an encryption computation. Therefore, a recent solution [58] avoids this by performing the cryptographic computations at the reader end, and store the resulting information in the tags. This solution of course requires careful modification of the reader-tag protocols. A number of cryptographic protocols for privacy protection of library RFID activity are discussed in [78]. Some of the cryptographic schemes [62, 69, 82] work with *re-writable* memory in the tags in order to increase security. The tags are encrypted, and the reader is able to decrypt them when they send them to the server, in order to determine the unique meta-information in the tag. The reader also has the capability to re-encrypt the tag with a different key and write it to its memory, so that the (encrypted) tag signal for an eavesdropper is different at different times. Such a scheme provides additional protection because of repeated change in the encrypted representation of the tag, and prevents the eavesdropper from uniquely identifying the tag at different times.

An interesting solution for making it difficult to read tags in an unauthorized way is the use of *blocker tags* [59, 60]. Blocker tags exploit the collision properties of RFID transmission, which are inherent in this technology. The key idea is that when two RFID tags transmit distinct signals to a reader at the same time, a broadcast collision occurs, which prevents the reader from deciphering either response. Such collisions are in fact very likely to occur during the normal operation of the RFID infrastructure. In order to handle this issue, RFID readers typically use anti-collision protocols. The purpose of blocker tags is to emit signals (or spam) which can defeat these anti-collision protocols, thereby causing the reader to stall. The idea is that blocker tags should be implemented in a way, that it will only spam unauthorized readers, thereby allowing the authorized readers to behave normally. Details of the blocking approach are discussed in [9].

It was inferred in [111] that the greater threat to privacy arises from the eavesdropping of signals sent from the reader (which can be detected much further away), rather than reading the tag itself (which can be done only at a much closer distance). In fact, the IDs being read by the tree-walking protocol can be inferred merely by listening to the signals being broadcast by the reader. Therefore, it has been proposed in [111] to encrypt the signals being sent by the reader in order to prevent privacy attacks by eavesdropping of *reader* signals.

It is also possible to modify RFID tags to cycle through a set of *pseudonyms* rather than emit a unique serial number [58]. Thus, the tag cycles through a set of k pseudonyms and emits them sequentially. This makes it more difficult for an attacker to identify the tags, because they may only be able to scan different pseudonyms of the tags at different times. Of course, if the attacker is aware of the method being used in order to mask the tag, they may try to scan the tag over a longer period of time, in order to learn all the pseudonyms associated with the tag. This process can be made more difficult for an attacker by increasing the time it takes for the tag to switch from one pseudonym to another.

5.2 Privacy in Data Sharing and Management

Since the functionality of the internet of things is based on the data communication between different entities, and the underlying data may often be person-centric, the ability to provide privacy during the data transmission and sharing process is critical. For example, in a mobile application, the GPS data for a user may be collected exactly, but may not necessarily be shared exactly. A variety of techniques may be used in order to reduce the privacy challenges during data sharing:

- Many applications may require only *aggregate* information collected by the sensors, rather than exact information about individuals. For example, traffic conditions in a vehicular sensing applications can be inferred with the use of aggregate data. Examples of systems which use aggregate data for privacy-preserving queries in smart vehicular sensing environments are discussed in [87].

- A variety of privacy-preservation mechanisms such as k-anonymity, ℓ-diversity, and t-closeness reduce the accuracy of the data before sharing it with other entities [10]. For example, for video data, the faces in the videos can be blurred in order to reduce the likelihood of identification [112]. In the context of mobile and location data, a variety of methods such as spatial cloaking, spatial delays, adding noise to locations etc. [29, 8] are incorporated in order to increase data privacy. A detailed discussion of methods for increasing location privacy are provided in [8].

In practice, it is desirable to set up a set of policies which can allow users to specify which kinds of data they would like to share about themselves. The W3C group has defined the *Platform for Privacy Preferences (P3P)* [125], which provides a language for description of privacy preferences. This allows the user to set specific privacy requirements, and also allows for automatic negotiation between the personal information needs of a user and their privacy preferences.

The issue of privacy has also been addressed in the context of the semantic web [38, 64]. The broad idea in [38] is that users are able to retain control over who has access to their personal information under different conditions. For instance, one may allow their colleagues to access their calender over the weekend, but not over weekdays. In addition, it is desirable to fine tune the granularity of the query responses, depending upon the identity of the person who is performing the queries. A semantic web architecture is proposed in [38], which supports the automated discovery and access of personal resources for a variety of context-aware applications. Each source of contextual information (e.g. a calendar, location tracking functionality, collections of relevant user preferences, organizational databases) is represented as a semantic web service. A semantic e-Wallet acts as a directory of contextual resources for a given user, while enforcing her privacy preferences. Privacy preferences enable users to specify what information can be provided to whom in different contexts. They also allow users to specify *obfuscation rules*, which control the accuracy or inaccuracy of the information provided in response to different queries under different conditions.

5.3 Data Security Issues

Since the data collection nodes in the internet of things spend a lot of time unattended, it opens up the system to a number of security threats. For example, *data integrity* is often a concern, because a malicious adversary can change the data at various stages in the pipeline. In order to address these issues, a number of methods have been designed to password-protect the writing of the memory in the RFID tags or the sensor nodes. A number of solutions for password protection in the context of sensor data are proposed in [4, 70]. For RFID data, this is a greater challenge because the password-protection process requires the use of energy-intensive cryptographic algorithms. This would require an onboard battery (active tag) for enablement, and larger energy consumption requirements are usually undesirable. In this context, a number of methods, which have low energy requirements for these cryptographic solutions in RFID have been proposed recently [26, 37].

The use of RFID technology also has a number of other security concerns. For example, RFID technology is highly dependent on the use of radio signals which are easily jammed. This can open the system to a variety of infrastructure threats, that can disrupt the data collection process. It has recently been demonstrated [19], that RFID tags can be cloned to emit the same identification code as another tag. This opens the system to fraud, when the RFID tag is used for the purpose of sensitive tasks such as payment, authentication or access control. As in the previous case, a number of cryptographic solutions are being proposed to increase the security of RFID technology [19].

A number of security issues also arise in the context of data representations on the semantic web. The data on the semantic web is dynamic and open, which makes it a challenge from a security perspective. Therefore, methods have been proposed for marking up web entities with a semantic policy language, and the use of distributed policy management as a tool for security [63]. The major challenge which is identified with implementing such security policies for the semantic web is the decentralized nature of the semantic web, with a large number of entities, each with its resources, services, agents, users, and their heterogeneity. The work in [63] proposes a distributed policy framework, in which every entity can specify their own policy, since there is no centralized policy. A policy language is proposed, based on RDF-S, in order to markup security information. The policies are specified in terms of *properties* of users, agents, services or resources, rather than *identities*, since full authentication is not possible on the web. A related privacy-preserving ontology framework, based on OWL-S, is proposed in [64].

6. Conclusions

The internet of things is a vision, which is currently being built. It is based on the unique addressability of a large number of objects which may be RFID-based tags, sensors, actuators, or other embedded devices, which can collect and transmit data in an automated way. The massive scale of the internet of things brings a number of corresponding challenges of scale in terms of IP-addressability, privacy, security, and data management and analytics. The internet-of-things has a long data-processing pipeline in terms of collection, storage, and processing, and the decisions made at the earlier stages of the pipeline can significantly impact the processing at later stages. Numerous research choices exist at the different stages of the pipelines, as is clear from the discussion in this chapter. This has lead to a fertile area for research, which is likely to remain of great interest to multiple communities of researchers over the next few years.

References

[1] D. J. Abadi, A. Marcus, S. R. Madden, K. Hollenbach. Scalable Semantic Web Data Management using vertical partitioning. *VLDB Conference*, 2007.

[2] D. J. Abadi, S. R. Madden, M. C. Ferreira. Integrating Compression and Execution in Column Oriented Database Systems. *SIGMOD Conference*, 2006.

[3] D. J. Abadi, D. S. Myers, D. J. DeWitt, S. R. Madden. Materialization Strategies in a Column-Oriented DBMS. *ICDE Conference*, 2007.

[4] R. Acharya, K. Asha. Data integrity and intrusion detection in wireless sensor networks, *Proceedings of the IEEE ICON*, 2008.

[5] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[6] C. C. Aggarwal. On Unifying Privacy and Uncertain Data Models, *ICDE Conference*, 2008.

[7] C. C. Aggarwal. Managing and Mining Uncertain Data, *Springer*, 2009.

[8] C. C. Aggarwal, T. Abdelzaher. Social Sensing. *Managing and Mining Sensor Data*, Springer, 2013.

[9] C. C. Aggarwal, J. Han. A Survey of RFID Data Processing, *Managing and Mining Sensor Data*, Springer, 2013.

[10] C. C. Aggarwal, P. S. Yu. Privacy-Preserving Data Mining, *Springer*, 2008.

[11] S. Alexaki, V. Christophides, G. Karvounarakis, G. Plexsoukis. On Storing Voluminous RDF Descriptions: The Case of Web Portal Catalogs, *WebDB*, 2001.

[12] S. Alexaki, V. Christophides, G. Karvounarakis, G. Plexsoukis, K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, *SemWeb*, 2001.

[13] R. Angles, C. Gutierrez. Querying RDF Data from a Graph Database Perspective, *ESWC*, 2005.

[14] K. Ashton. That 'Internet of Things' Thing. In: *RFID Journal*, 22 July, 2009.

[15] M. Atre, V. Chaoji, M. J. Zaki, J. Hendler. Matrix "Bit" loaded: A Scalable Lightweight Join Query Processor for RDF Data, *WWW Conference*, 2010.

[16] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A Survey, *Computer networks*, 54(16), pp. 2787–2805, 2010.

[17] M. Balazinska et al. Data Management in the World Wide Sensor Web, *Pervasive Computing*, April–June, 2007.

[18] D. Beckett. The Design and Implementation of the Redland RDF Application Framework. *WWW Conference*, 2001.

[19] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, M. Szydylo. Security Analysis of a Cryptographically Enabled RFID Device, *USENIX Security*, 2005.

[20] V. Bonstrom, A. Hinze, H. Schweppe. Storing RDF as a graph. *LA-WEB*, 2003.

[21] A. Broring et al. New Generation Sensor Web Enablement, *Sensors*, 11(3), 2011.

[22] M. Buettner, B. Greenstein, A. Sample, J.R. Smith, D. Wetherall. Revisiting smart dust with RFID sensor networks, *Proceedings of ACM HotNets*, 2008.

[23] C. Bornhovd, T. Lin, S. Haller, J. Schaper. Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-Id Infrastructure, *VLDB Conference*, 2004.

[24] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. *ISWC*, 2002.

[25] V. Bychkovskiy, S. Megerian, D. Estrin, M. A. Potkonjak. collaborative approach to in-place sensor calibration. *IPSN Conference*, 2003.

[26] B. Calmels, S. Canard, M. Girault, H. Sibert. Low-cost cryptography for privacy in RFID systems, *Proceedings of IFIP CARIDS*, 2006.

[27] F. Chang et al. Bigtable: A Distributed Storage System for Structured Data, *OSDI*, 2006.

[28] H. Chen, W.-S. Ku, H. Wang, M.-T. Sun. Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing. *ACM SIGMOD Conference*, 2010.

[29] C.-Y. Chow, M. F. Mokbel. Privacy of Spatial Trajectories. *Computing with Spatial Trajectories*, pp. 109–141, 2011.

[30] I. Constandache, R. Choudhury, I. Rhee. Towards mobile phone localization without war-driving, *INFOCOM Conference*, 2010.

[31] D. Cook, L. Holder. Sensor selection to support practical use of health-monitoring smart environments. *Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery* 1(4): pp. 339–351, 2011.

[32] R. Cyganiak. A Relational Algebra for SPARQL, *HP-Labs Technical Report, HPL-2005-170.* `http://www.hpl.hp.com/techreports/2005/HPL-2005-170.html`.

[33] J. Dean, S. Ghemawat. MapReduce: A flexible data processing took, *Communication of the ACM*, Vol. 53, pp. 72–77, 2010.

[34] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong. Model-driven data acquisition in sensor networks. *VLDB*, 2004.

[35] L. Ding et al. Swoogle: A Semantic Web and Metadata Search Engine, *ACM CIKM Conference*, 2004.

[36] M. Schmitter-Edgecombe, P. Rashidi, D. Cook, L. Holder. Discovering and Tracking Activities for Assisted Living, *The American Journal of Geriatric Psychiatry*, In Press, 2011.

[37] M. Feldhofer, S. Dominikus, J. Wolkerstorfer. Strong authentication for RFID systems using AES algorithm, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, 2004.

[38] F. Gandon, N. Sadeh. Semantic Web Technologies to Reconcile Privacy and Context Awareness, *Web Semantics: Science, Services and Agents on the Worldwide Web*, 1(3), pp. 241–260, 2004.

[39] A. Gangemi. DOLCE UltraLite OWL Ontology, `http://www.loa-cnr.it/ontologies/DUL.owl`, 2007

[40] S. L. Garfinkel, A. Juels, R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions, *IEEE Security and Privacy*, 3(3), 2005.

[41] D. Giusto, A. Iera, G. Morabito, L. Atzori (Eds.), The Internet of Things, *Springer*, 2010.

[42] H. Gonzalez, J. Han, X. Li, D. Klabjan. Warehousing and Analyzing Massive RFID Data Sets. *ICDE Conference*, 2006.

[43] T. Gruber. A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 2(5), pp. 199–200, 1993.

[44] R. Guha, R. McCool, E. Miller. Semantic Search, *WWW Conference*, 2003.

[45] D. Guinard, V. Trifa. Towards the Web of Things: Web Mashups for Embedded Devices, *WWW Conference*, 2009.

[46] D. Guinard, V. Trifa, F. Mattern, E. Wilde. From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices, *Architecting the Internet of Things*, Springer, 2011.

[47] A. Gupta, M. Srivasatava. Developing auto-id solutions using sun java system rfid software, October 2004.
`http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html`

[48] J. Han, J.-G. Lee, H. Gonzalez, X. Li. Mining Massive RFID, Trajectory, and Traffic Data Sets (Tutorial). *ACM KDD Conference*, 2008.
Video of Tutoral Lecture at: `http://videolectures.net/kdd08_han_mmrfid/`

[49] S. Harris, N. Gibbins. Efficient Bulk RDF Storage. *PSSS*, 2003.

[50] S. Harris, N. Shadbolt. SPARQL query processing with conventional relational database systems. *SSWS*, 2005.

[51] A. Harth, S. Decker. Optimized index structures for querying RDF from the web. *LA-WEB*, 2005.

[52] O. Hassanzadeh, A. Kementsietsidis. Data Management Issues for the Semantic Web, *ICDE Conference*, 2012.

[53] J. Hayes, C. Gutierrez. Bipartite graphs as intermediate model for RDF. *ISWC*, 2004.

[54] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. A pipelined framework for online cleaning of sensor data streams. *ICDE Conference*, 2006.

[55] S. R. Jeffrey, M. Garofalakis, M. J. Franklin. Adaptive Cleaning for RFID Data Streams, *VLDB Conference*, 2006.

[56] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. Declarative Support for RFID Data Cleaning, *Pervasive*, 2006.

[57] S. Jirka, A. Broring, C. Stasch. Discovery Mechanisms for the Sensor Web,*Sensors*, 9, pp. 2661–2681, 2009.

[58] A. Juels. Minimalist Cryptography for RFID Tags. *Conference on Security in Communication Networks*, 2004.

[59] A. Juels, R. Rivest, M. Szydlo. The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy. *ACM Conference on Computer and Communication Security*, pp. 103–111, 2003.

[60] A. Juels, J. Brainard. Soft Blocking: Flexible Blocker Tags on the Cheap, *Workshop on Privacy in the Electronic Society (WPES 04)*, pp. 1–7, 2004.

[61] A. Juels. RFID Security and Privacy: A Research Survey, *IEEE Journal on Selected Areas in Communication*, vol. 24, pp. 381–394, Feb. 2006.

[62] A. Juels, R. Pappu. Squealing Euros: Privacy protection in RFIDenabled banknotes. *Proceedings of Financial Cryptography*, Springer–Verlag, 2003.

[63] L. Kagal, T. Finin, A. Joshi. A Policy-based Approach to Security for the Semantic Web, *ISWC*, 2003.

[64] L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T. Finin, K. Sycara. Authorization and Privacy for Semantic Web Services, *IEEE Intelligent Systems*, 19(4), 2004.

[65] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, V. Terziyan. Smart Semantic Middleware for the Internet of Things, *ICINCO*, 2008.

[66] N. Khoussainova, M. Balazinska, D. Suciu. Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. *Fifth International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06)*, 2006.

[67] C. Kiefer, A. Bernstein, M. Stocker. The fundamentals of iSPARQL – a virtual triple approach for similarity-based Semantic Web tasks. *ISWC*, 2007.

[68] Y. Kim, B. Kim, J. Lee, H. Lim. The path index for query processing on RDF and RDF Schema. *ICACT*, 2005.

[69] S. Kinoshita, F. Hoshino, T. Komuro, A. Fujimura, M. Ohkubo. Low-cost RFID privacy protection scheme. *IPS Journal*, 45(8), 2004.

[70] R. Kumar, E. Kohler, M. Srivastava. Harbor: software-based memory protection for sensor nodes, *IPSN Conference*, 2007.

[71] M. Langheinrich. A Survey of RFID Privacy Approaches. *Personal and Ubiquitous Computing*, Springer, 2008.

[72] Y. Lei, V. Uren, E. Motta. SemSearch: A Search Engine for the Semantic Web, *Managing Knowledge in a World of Networks*, 2006.

[73] T. B. Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientific American*, 2001.

[74] A. Madan, S. Moturu, D. Lazer, A. Pentland. Social Sensing: Obsesity, Healthy Eating and Exercise in Face-to-Face Networks, *Wireless Health*, 2010.

[75] A. Matono, T. Amagasa, M. Yoshikawa, S. Uemura. A path-based relational RDF database. *ADC*, 2005.

[76] E. Miller. An Introduction to the Resource Description Framework, *D-Lib Magazine*, 4(5), 1998.

[77] P. Mika, G. Tummarello. Web semantics in the clouds. *IEEE Intelligent Systems*, 23(5), pp. 82–87, 2008.

[78] D. Molnar, D. Wagner. Privacy and Security in Library RFID: Issues, Practices and Architectures, *CCS*, 2004.

[79] T. Neumann, G. Weikum. Scalable Join Processing on Very Large RDF Graphs, *ACM SIGMOD Conference*, 2009.

[80] T. Neumann, G. Weikum. The RDF-3X Engine for Scalable Management of RDF Data, *VLDB Journal*, 19(1), 2010.

[81] M. Ohkubo, K. Suzuki, S. Kinoshita. RFID Privacy Issues and Technical Challenges, *Communications of the ACM*, 48(9), 2005.

[82] M. Ohkubo, K. Suzuki, S. Kinoshita. A cryptographic approach to "privacy-friendly" tags. *RFID Privacy Workshop*, 2003.

[83] C. Olston, B. Reed, U. Srivastava, R. Kumar. PigLatin: A Not so Foreign Language for Data Processing, *ACM SIGMOD Conference*, 2008.

[84] J. Paek, J. Kim, R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones, *MobiSys*, 2010.

[85] Z. Pan, J. Heflin. DLDB: Extending relational databases to support Semantic Web queries. *PSSS*, 2003.

[86] J. Perez, M. Arenas, C. Gutierrez. Semantics and Complexity of SPARQL, *ISWC*, 2006.

[87] R. A. Popa, H. Balakrishnan, A. Blumberg. VPriv: Protecting Privacy in Location-Based Vehicular Services. *USENIX Security Symposium*, 2008.

[88] E. Prud'hommeaux, A. Seaborne. SPARQL Query Language for RDF (Working Draft), `http://www.w3.org/TR/2007/WD-rdf-sparql-query-20070326/`, 2007

[89] D. Anicic, P. Fodor, S. Rudolph, N. Stojanovic. EP-SPARQL: a unified language for event processing and stream reasoning, *WWW Conference*, 2011.

[90] B. Quilitz, U. Leser. Querying Distributed RDF Data Sources with SPARQL, *The Semantic Web: Research and Applications*, 2008.

[91] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, M. J. Neely. Energy-delay tradeoffs in smartphone applications, *MobiSys*, 2010.

[92] R. Raskin, M. Pan, Semantic Web for Earth and Environmental Terminology (SWEET, *Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[93] S. E. Sarma, S. A. Weis, D.W. Engels. Radio-frequency identification systems. *CHES*, pp. 454–469, 2002.

[94] S. E. Sarma, S. A. Weis, D.W. Engels. RFID systems, security and privacy implications. *Technical Report MIT–AUTOID–WH–014*, AutoID Center, MIT, 2002.

[95] M. Schmidt, M. Meier, G. Lausen. Foundations of SPARQL Query Optimization, *ICDT Conference*, 2010.

[96] A. Sheth, C. Henson, S. Sahoo. Semantic Sensor Web, *IEEE Internet Computing*, 2008.

[97] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, D. Reynolds. SPARQL basic graph pattern optimization using selectivity estimation. *WWW Conference*, 2008.

[98] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madde, E. O'Neil, P O'Neil, A. Rasin, N. Tran, S. Zdonik. C-Store: A Column-Oriented DBMS, *VLDB Conference*, 2005.

[99] B. Sterling. Shaping Things – Mediawork Pamphlets, *The MIT Press*, 2005.

[100] W. J. Tu, W. Zhou, S. Piramuthu. Identifying RFID-embedded objects in Pervasive Healthcare Applications, *Decision Support Systems*, 46(2), 2009.

[101] T. Tudorache, J. Vendetti, N. F. Noy. Web-Protege: A Lightweight OWL Ontology Editor for the Web, *OWLED*, 2008.

[102] G. Tummarello, R. Delbru, E. Oren. Sindice.com: Weaving the Open Linked Data, *The Semantic Web*, 2007.

[103] O. Udrea, A. Pugliese, V. Subrahmanian. GRIN: A Graph Based RDF Index. *AAAI*, 2007.

[104] J. Urbani, S. Kotoulas, E. Oren, F. van Harmelen. Scalable Distributed Reasoning using MapReduce, *The Semantic Web - ISWC*, 2009.

[105] J. Urbani, J. Maassen, H. Bal. Massive Semantic Web data compression using MapReduce, *HPDC*, 2010.

[106] R. Want. An Introduction to RFID Technology, *IEEE Pervasive*, 2006.

[107] R. Want. Enabling Ubiquitous Sensing with RFID, *Computer*, 37(4), pp. 84–86, 2004.

[108] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, G. Borriello. Building the Internet of Things Using RFID. *IEEE Internet Computing*, 13(3), May–June, 2009.

[109] N. Wiegand, G. Berg-Cross, D. Varanka. Proceedings of the 2011 International Workshop on Spatial Semantics and Ontologies, *SSO* 2011.

[110] C. Weiss, P. Karras, A. Bernstein. Hexastore: Sextuple Indexing for Semantic Web Data Management, *VLDB Conference*, 2008.

[111] S. A. Weis, S. Sarma, R. Rivest, D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. *First International Conference on Security in Pervasive Computing*, 2003.

[112] J. Wickramasuriya, M. Datt, S. Mehrotra, N. Venkatasubramanian. Privacy-protecting data collection in Media Spaces, *ACM Multimedia Conference*, 2004.

[113] K. Wilkinson. Jena property table implementation. *SSWS*, 2006.

[114] K. Wilkinson, C. Sayers, H. A. Kuno, D. Reynolds. Efficient RDF storage and retrieval in Jena2. *SWDB*, 2003.

[115] T. White. Hadoop: The Definitive Guide. *Yahoo! Press*, 2011.

[116] D. Wood, P. Gearon, T. Adams. Kowari: A platform for Semantic Web storage and analysis. *XTech*, 2005.

[117] Z. Zhuang, K.-H. Kim, J. P. Singh. Improving energy efficiency of location sensing on smartphones, *MobiSys*, 2010.

[118] The EPCglobal Architecture Framework, March 2009. `http://www.epcglobalinc.org`

[119] Semantic Sensor Network XG Final Report, `http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/`, 2011.

[120] iAnyWhere Solutions Inc Whitepaper: *Manage Data Successfully with RFID Anywhere Edge Processing*, `http://www.sybase.com/files/White_Papers/SybaseRFID_edgepro-053107-wp.pdf`.

[121] http://seattle.intel-research.net/wisp/

[122] http://www.ipso-alliance.org/

[123] http://www.w3.org/TR/rdf-primer/#rdfmodel

[124] http://www.ted.com/talks/sebastian_thrun_google_s_driverless_car.html

[125] http://www.w3.org/TR/P3P11/

[126] http://hadoop.apache.org/hbase

[127] http://www.sindice.com

[128] http://sw.deri.org/2007/07/sitemapextension

[129] http://swoogle.umbc.edu/

Chapter 13

A SURVEY OF DATA MINING METHODS FOR SENSOR NETWORK BUG DIAGNOSIS

Tarek Abdelzaher
Department of Computer Science
University of Illinois at Urbana Champaign
zaher@cs.illinois.edu

Jiawei Han
Department of Computer Science
University of Illinois at Urbana Champaign
hanj@cs.illinois.edu

Abstract This chapter surveys recent debugging tools for sensor networks that are inspired by data mining algorithms. These tools are motivated by the increased complexity and scale of sensor network applications, making it harder to identify root causes of system problems. At a high level, debugging solutions in the domain of sensor networks can be classified according to their goal into two distinct categories; (i) solutions that attempt to localize errors to a single node, component, or code snippet, and (ii) solutions that attempt to identify a global pattern that causes misbehavior to occur. The first category inherits the usual wisdom that problems are often localized. It is unlikely for independent failures to coinside. Hence, while many different trouble symptoms may occur simultaneously, they typically arise from a single misbehaving component such as a failed radio or a crashed node that may, in turn, trigger a cascade of other problems. In contrast, the second category of solutions is motivated by interactive complexity problems. They seek to uncover bugs in networked sensing systems that arise due to unexpected interactions between components. The underlying assumption is that individual components are easier to test, which ensures that they work well in isolation. Therefore, practical software systems seldom fail due to a single poorly-coded component. Rather, they fail due to an unexpected interaction pattern between *individually well-behaved* components. The

challenge is to uncover the global interaction patterns that leads to the problem, as opposed to chasing a local root cause. The chapter describes the above different techniques and concludes with a brief review of other troubleshooting work, not inspired by data mining literature.

Keywords: Sensor networks, debugging, data mining, interactive complexity.

1. Introduction

The growing size and complexity of sensor networks makes troubleshooting them an increasingly difficult undertaking. There are fundamentally two ways to ensure system correctness: either ensure absence of faults by design, or develop techniques to detect and troubleshoot them at run-time.

In the broader embedded systems domain (of which sensor networks are a more recent subcategory), the predominant approach for ensuring correctness has been to prove absence of bugs by design. Distributed embedded systems and protocols can typically be described by state automata, where vertices represent logical states and edges represent state transitions. Such transitions may be triggered, for example, by events in the environment, occurrence of particular input values, or different communication events. Sensor network tools such as FSMGen [54] were developed to automatically construct an approximate finite state machine model of distributed sensing programs.

The approach allows one to reason about states that may be "bad" (i.e., represent different types of failures or undesirable behavior). These states may be reached through a particular confluence of events that trigger the "right" pattern of state transitions, leading to the bad state (i.e., a bug manifestation). A significant amount of embedded system literature focused on analyzing *reachability* of bad states in the software system's state machine. Given a model of the system, if a particular bad state is reachable from the current state, then the system is vulnerable in that there exists a sequence (or, more generally, a pattern) of events that may cause the bad behavior to manifest. A typical model-checking tool [5] performs reachability analysis to determine if certain bad states can occur, and either proves them to be unreachable or returns an example of the bug-causing pattern. Instances of recent system design techniques in embedded computing that provably avoid bad states can be found in domains as diverse as avoinics software [4], autonomous ground vehicles [7, 8], collision avoidance systems [20, 19], and medical pacemakers [6], to name a few.

Unfortunately, when applied to sensor networks, today's formal methods tools for embedded software are greatly hampered by the dimentionality explosion that arises from massive concurrency. Different event interleavings at different nodes can generate an exponential number of possible states, making analysis intractable. While a substantial amount of work does address formal correctness assurances in sensor network systems, current solutions fall short of a complete correctness proof. For example, they might feature *approximate* state coverage, focus on verifying *individual* components, consider *single-node* systems, or *simplify semantics* of communication and sensing.

To appreciate the need for *run-time* troubleshooting, which is the topic of this survey, it is informative to consider the limitations of present formal *pre-run-time* approaches. A significant set of pre-run-time tools focus on checking models of application code written for popular sensor network operating systems, such as TinyOS [59], to ensure that certain desirable properties are never violated. Early work [91, 72] sought abstract high-level models for TinyOS applications. The correspondence between these models and actual software implementation, however, would remain to be verified. Formal method-based solutions were also proposed to verify individual components or protocols, such as security protocols in [42] (expressed in the HLPSL [16] high-level formal specification language), sensor coverage protocols in [74] (expressed in real-time Maude [73]), and others [24]. To reduce the searched state space, approximations and incomplete coverage were also considered. For example, T-check [60] is a model checker that explores a simplified search space (of a protocol simulation), in order to check safety and liveness properties. KleeNet [80, 79] takes the alternative approach of symbolic execution, where given symbolic inputs, all paths of a distributed program are traversed in search for bugs. However, not all combinations of possible timings of distributed race conditions are considered.

More recent approaches verify safety assersions of actual complete application code [11, 100, 71]. For example, recent work on compile-time checking [100] verifies distributed sensor network programs directly in NesC [31] (a common sensor network programming language), given a simplified communication and sensing model. Another compile-time checker, called Anquiro [71], maps C programs written for Contiki [25] (also a popular sensor network operating system) to a state space where they can be statically verified. While these systems can uncover bugs in real code, a general problem with them is that the state space grows exponentially with the amount of communication and race conditions. The intractability of comprehensive solutions for uncovering sensor network bugs at design time increases the importance of *run-time troubleshooting*

when unknown bugs manifest themselves after deployment, thus motivating the work surveyed in this chapter.

In view of the above, a number of automated techniques were recently developed for troubleshooting sensor networks after deployment in order to identify causes of anomalous behavior, recover from problems, and reduce ownership costs. An important category of these techniques leverage data mining literature on identification, classification, and understanding of complex patterns in large, highly coupled systems [92, 30, 97, 43, 39, 41, 21, 84] with applications ranging from biological processes [77] to commercial databases [68]. Data mining techniques help the discovery of hidden patterns that may be responsible for software malfunction.

While the use of data mining in network troubleshooting is promising, it is by no means a straightforward application of existing techniques to a new problem. Networked software execution patterns are not governed by "laws of nature", DNA, business transactions, or social norms. They are limited only by programmers' imagination. The increased diversity and richness of software interaction patterns make it harder to zoom-in on potential causes of problems without embedding some knowledge of networking, programming, and debugging into the data mining engine. This chapter describes cross-cutting solutions that leverage the power of data mining to uncover hard-to-find bugs in distributed sensing systems.

There are two fundamentally different schools of thought when it comes to using data mining tools for designing debugging solutions. The first one adopts the belief that problems in large systems are inherently localized. It is uncommon for independent failures to coincide. Hence, when trouble occurs, while symptoms may be many, the challenge is to find the single root cause (or the smallest set of independent causes) that can trigger the observed avalanche of problems. Finding this single root cause can be cast as a classification problem, in which leafs of the classifer are the different diagnostic answers. A challenge is to determine the rules or features that can reliably discriminate between the different root cause failure scenarios. These approaches are covered in Section 2.

The second school of thought argues that in professional production systems (that are well-designed and well-maintained) most failures arise due to *unexpected interactions* between components. Individual components are designed to high-standards and seldom misbehave on their own. It is the large combination of such components that can result in subtle problems because of interactions that may have not been envisioned at design time. Hence, the debugging tool should be looking for an interaction pattern, such as a particular sequence or a particular

graph of events that causes failure (rather than a single component to blame). These approaches are described in Section 3.

Finally, for completeness, we briefly survey in Section 4 the remaining debugging techniques in sensor networks that do not use data mining tools.

2. Classification-based Bug Localization

A popular category of diagnostic tools in sensor networks, that takes inspiration from data mining and machine learning literature, is based on classifiers. These tools label the current state of the network as either normal or abnormal, and recognize different types of abnormal behavior. The used classifiers typically fall into one of three general types presented in the subsections below; namely, *simple rule-based classifiers* (where classification rules are input by domain experts), *supervised classifiers* (that need examples of good and bad behavior to learn to recognize bugs), and *unsupervised classifiers* that learn to detect anomalous behavior and relate the anomaly to small pieces of code that are easy to inspect manually.

2.1 Simple Rule-based Classifiers

One of the earliest classification-based approaches in sensor network troubleshooting literature was implemented in a tool, called Sympathy [78] that uses a set of prespecified rules for root cause analysis and failure source localization. The tool diagnoses root causes of communication problems. In this framework, nodes proactively exchange diagnostic information such as connectivity and flow states. When nodes receive fewer messages than normal, they run a diagnostic tree fault-classification algorithm that utilizes received information to localize the problem. A contribution of the tool lies in understanding the most probable cause of failure among different alternative explanations. For example, a base-station that collects data from a sensor network is trivially aware of the reachability of all nodes. When a number of nodes suddenly stop reporting, it is not immediately clear what the reason is. It could, for example, be attributed to the failure of only one node that is the parent of the disconnected subtree in the data collection tree. By inspecting node connectivity information (e.g., previously received node neighbor tables), Sympathy is able to reason about such failures and come up with the most likely explanation of the primary failure. For example, it could identify which particular node died (which is the primary failure), isolating an entire subtree (which is a secondary, dependent failure), as opposed to suspecting the whole subtree of dying.

A lighter passive version of this approach, called PAD [65], is described in more recent literature. PAD uses a variant of belief networks to infer the most likely cause of communication misbehavior symptoms. It is a passive approach in that the diagnostic system does not introduce communication traffic of its own, but rather piggybacks status bytes on existing traffic. It is light in that the system restricts network status monitoring overhead to only two bytes per packet. These bytes, called a *mark*, simply carry the ID of some intermediate node on the packet's path to the collection base-station, as well as the hop count from the intermediate node to the source. By cleverly decising which node should mark which packet, network topology information can be reconstructed at the base-station, and diagnostics can be performed to identify the likely causes when anomalous changes occur.

2.2 Supervised Classifiers

Two different examples of supervised learning approaches in sensor network debugging literature are the Diagnostic PowerTracer [50] and the SNTS debugging tool [51]. The former learns signatures of different *known failures* reproduced intentionally during laboratory testing, such that it could later recognize these failures in the field. The latter learns the distinguishing features of newly encountered *unknown failures* by contrasting them with normal behavior. These features then provide clues regarding the root cause.

The Diagnostic PowerTracer was motivated by the need to remotely identify root causes of failures of *silent nodes*. While techniques that localize the failed node [78, 65] can pinpoint which node died, they offer less insight into the reason for its silence after the node stops responding. To circumvent this challenge, the PowerTracer features an external hardware module that samples (at the rate of a few Hz) the power consumed by each sensor node, and sends those samples via a low-power low-bandwidth radio to a diagnostic base-station. The base-station quantizes the reported power values in the trace into discrete symbols (called power consumption *states*). It then derives from the sequence of symbols in the trace a probabilistic state transition diagram and matches it against those of known failures to identify the problem with the silent node. The approach was shown to correctly differentiate between OS crashes, radio failures, antenna failures, water-induced electrical failures, and battery depletion, among other failure states.

A different supervised classification-based approach is exemplified in the SNTS debugging tool [51]. The tool uses the PART [28] algorithm from the Weka data mining library [70, 36] as the underlying classifer to

distinguish conditions associated with bug manifestations. The input to the algorithm is the set of messages collected when network behavior is correct and the set collected when behavior is anomalous. Messages are labeled accordingly. Their headers are inspected. Content of different header fields (such as message type) constitutes the different variables, or dimensions, for classification. The counts of how many times messages of different types were received also constituted classification dimentions.

As an example of its applicability, the aforementioned algorithm was used to troubleshoot EnviroTrack [1], a distributed target tracking protocol for sensor networks. EnviroTrack was designed to detect intruders who cross a sensor network field and track their movements. It occasionally generated spurious target IDs, causing the number of detected targets to be larger than the actual number of targets in the field. During debugging, the system was tested by running targets through the sensor network and recording all messages communicated between nodes. The number of targets reported by the sensor network was also recorded. If this number was correct, the collected message logs were labeled "good". Otherwise, they were labeled "bad". The SNTS tool analyzed these logs, revealing that a surprising 80% of all failures to maintain the correct target count were correctly predicted by the classifier using a single rule; namely, the absence of messages of type *member-to-leader* from the log.

The designer of the target tracking protocol explained that sensors, who see the target, form a group and elect a leader among themselves. Members of this group continue communicating with the leader through *member-to-leader* messages. As the target moves, membership of the group changes (as the nodes that see the target change). Leader hand-off eventually occurs to make sure the leader is always close to the target. Absence of member-to-leader messages therefore means that the target is seen by only one sensor at a time. Hence, that sensor is the leader. There are no further group members, and therefore no member-to-leader messages. This, it turns out, was precisely the problem with EnviroTrack. Leader hand-off in EnviroTrack failed when the leader was the only node who could see the target, since there were no other nodes to hand off the target to. A consequence of a failed hand-off was that new sensors that eventually detect the target would assign it a different new ID, thinking it is a new target. The above illustrates how the failure condition uncovered by the tool led to a diagnosis of the problem. Notice that the tool itself does not have the knowledge to relate the observed problem to lines in the code that explain how it happens. However, by identifying other conditions that are correlated with the occurrence of the specific failure mode, it helps the designer recognize what triggers the problem.

2.3 Unsupervised Classifiers

In contrast to the above, an entirely unsupervised approach is adopted in Sentomist [101], a tool that focuses on profiling event handlers. Sentomist runs on top of Avrora [86], a sensor network emulator with a complete emulation of common sensor network hardware platforms such as Mica2 motes[1] (based on the ATMega128L processor and the CC1000 radio). An extension to Avrora is used to count the number of different types of instructions executed in event handlers. The vectors of such counts for each type of handler are then input to an anomaly detection routine that implements a one-class SVM algorithm [90]. The algorithm determines which counts are statistically "normal", and flags handlers with abnormal counts. Those handlers are then subjected to manual inspection. The idea behind this method lies in the assumption that the code of buggy handler instances must be different from that of correctly executing ones. Hence, looking for anomalous code execution patterns can likely lead to locations of bugs. The protocol was used to identify bugs in sensor networks such as abnormal packet losses during multi-hop data forwarding, and unhandled race conditions between different protocols.

A particularly interesting case study in applying Sentomist [101] was one where its authors emulated a combination of a heartbeat message exchange protocol and the Collection Tree Protocol (CTP) [33]. They focused on profiling the timer event handler. The anomaly detection routine ranked the different invocations of the handler by the degree to which their instruction count vectors deviated from normal behavior. The top ranked handler instances were then inspected for bugs. Indeed, code analysis revealed that the 4th ranked handler instance exhibits a race condition where CTP fails to transmit due to contention with the heartbeat message exchange protocol. The contention is apparently not handled correctly, causing the "busy" bit of the underlying communication device not to be cleared. Hence, all subsequent communication fails. The bug was uncovered by manual inspection of the particular execution sequence of the handlers flagged by Sentomist.

The above has been a quick tour through represenative examples of different classification-based debugging techniques whose purpose was to *localize* bugs. In other words, they aimed to single-out a node or component that is responsible for anomalous behavior. Next, we describe techniques that adopt an entirely different methodology in searching for

[1] A list of common sensor network platforms and their documentation is maintained at http://www.tinyos.net/scoop/special/hardware

root causes of execution problems. Namely, they attempt to uncover underlying *interaction* problems.

3. Troubleshooting Interactive Complexity

A number of recent analysis techniques in sensor network debugging literature aim to uncover root causes of errors resulting from anomalous *interactions* among large numbers of components. Modern networked sensing systems feature heterogeneity and tight interactions between computation, communication, sensing, and control. Tight interactions breed interactive complexity; the primary cause of failures and vulnerabilities in complex systems [76]. While individual devices and subsystems may operate well in isolation, their composition might result in incompatibilities, anomalies or failures that are typically very difficult to troubleshoot. On the other hand, software re-use is impaired by the customized nature of application code and deployment environments, making it harder to amortize debugging, troubleshooting, and tuning cost.

Techniques discussed in this section are grounded in the assumption that today's professional developers are very good at debugging *individual components*. Hence, in large systems, unresolved problems arise not from individual component failures, but from unexpected complex interactions that evade developer imagination (and defeat scalability limits of pre-run-time analysis tools).

Data mining offers a solution to the troubleshooting problem by empirically uncovering those event sequences or patterns that led to the bad states. Rather than exploring the entire space of possible states, data mining solutions use efficient pruning techniques to focus the search on those patterns that are correlated with anomalous behavior. The data mining approach further exhibits two important qualities that makes it suitable for the debugging techniques presented in this section:

- *Exploiting non-reproducible behavior:* Most hard-to-find bugs are hard to reproduce. Data mining approaches are good at exploiting non-determinism to improve understanding of system behavior. For example, discriminative mining requires examples of both good and bad system behavior to be able to isolate conditions correlated with good and bad. Non-reproducible bugs are thus inherently suited for analysis using data mining approaches as the lack of reproducibility itself and the inherent system non-determinism improve the odds of occurrence of sufficiently diverse behavior examples to help the troubleshooting system understand the relevant correlations and identify causes of problems.

- *Exploiting interactive complexity:* Interactive complexity describes a system where scale and complexity cause components to interact in unexpected ways. A failure that occurs due to such unexpected interactions is therefore hard to "blame" on any single component. This fundamentally changes the objective of a troubleshooting tool from finding a localized error, such as an incorrect pointer reference), to aiding with diagnosing a *pattern of events* (component interactions) that leads to a failure state. Various flavors of pattern mining algorithms, such as frequent pattern mining [38, 87], frequent subsequence mining [23], direct discriminative pattern mining [15], and numerical discriminative pattern mining [52] are particularly suited as the core analytic engine behind diagnostic debugging of anomalous interactions.

Conceptually, the techniques discussed below log a large number of events, correlate event patterns with manifestations of undesirable behavior, then find patterns that may be causally responsible for the failure. We classify these debugging techniques by the complexity of the patterns they support. We first describe finding *sequential patterns* (i.e., event sequences) correlated with bug manifestations. Next, we present techinques that find *event subgraphs* correlated with failures. Finally, generalizations to *symbolic patterns* are presented.

3.1 Sequence Mining

An example of a troubleshooting tool based on sequence mining is *Dustminer* [49]. It exploits the observation that, in a distributed wireless sensor network, certain sequences of events may lead to an undesirable or invalid state, causing the system to fail or exhibit poor performance. Hence, in principle, if one logs different types of events in the network, one may be able to use a form of discriminative sequence mining to capture sequences that lead to failure. As stated before, logs of runtime events are separated into two piles; a "good" pile, which contains the parts when the system performs as expected, and a "bad" pile, which contains the parts when the system exhibits bad behavior. A discriminative frequent pattern mining algorithm then looks for patterns (sequences of events) that exist with very different frequencies in the two piles. The tool used (a variation of) the Apriori algorithm [2] to find frequent patterns in each pile. Some of the more efficient algorithms for frequent itemset mining include FPgrowth [40] and PrefixSpan [75], but they do not handle gapped subsequence mining. It was shown that the basic Apriori algorithm could be extended in five ways to support the troubleshooting context:

- *Preventing false frequent patterns:* A basic gapped sequence mining algorithm considers all possible combinations of frequent subsequences of the original sequence with no regard to the amount of separation between individual events. As a result, it may generate subsequences combining events that are "too far apart" to be causally related. To alleviate this problem, the algorithm was extended to identify loops in the underlying program. Sequences were not allowed to span different loop iterations, localizing the search to the span of a single iteration of a program loop. A *dynamic search window* scheme was developed, where the first item of any candidate sequence was used to determine the search window. The window extended only until the next occurrence of this same item, and sequences were searched for only within individual windows. Patterns that cross window boundaries were not considered.

- *Suppressing redundant subsequences:* When frequent patterns were found, where one is a subsequence of the other, only the longest pattern was kept. This had impact on the ability to come up with discriminative patterns. The rule makes sense in debugging because not all subsets of "good" sequences are good. Forgetting a step in a multi-step procedure may well cause a failure. Hence, subsequences of good sequences could be bad. If the sequence `a b c d` occurs frequently during normal operation, and sequence `a c d` occurs frequently when a bug manifests, one must make sure that discriminative mining identifies the latter pattern as indeed discriminative despite the fact that it also occurs (as a subset of the frequent pattern `a b c d`) in the "good" case. Otherwise, failures that arise due to omission of some step (e.g., omission of `b` above) would not be found. Closed item set mining [89, 99] is commonly used in data mining to eliminate frequent subsets of a super set. Using this approach, sequences that are a subsequence of a longer pattern with similar support are eliminated.

- *Two-stage mining for infrequent events:* In debugging, sometimes less frequent patterns could be more indicative of the cause of failure than the more frequent patterns. A single mistake can cause many instances of damage. For example, a single node reboot event can cause a large number of message losses. In such cases, the most frequent patterns may not include the real cause of the problem. Fortunately, in the case of sensor network debugging, a solution may be inspired by the nature of the problem domain. The fundamental issue to observe is that much computation in sen-

sor networks is *recurrent*. Code repeatedly visits the same states (perhaps not strictly periodically), repeating the same actions over time. Hence, a single problem, such as a node reboot or a race condition that pollutes a data structure, often results in multiple manifestations of the same unusual symptom (such as multiple subsequent message losses or multiple subsequent false alarms). Catching these recurrent symptoms by a frequent pattern mining algorithm is much easier due to their larger frequency. With such symptoms identified, the search window can be narrowed to the neighborhood of those frequent patterns, and it becomes easier to correlate them with other less frequent preceding event occurrences.

- *Event frequencies and sampling:* One difficulty in debugging sensor network software is that the amount of logged events and the corresponding frequency of patterns can be different from run to run depending on factors such as length of execution and system load. A higher sampling rate at sensors, for example, may generate more messages and cause more events to be logged. Many logged event patterns in this case will appear to be more frequent. This is problematic when it is desired to compare the frequency of patterns found in "good" and "bad" data piles for purposes of identifying those correlated with bad behavior. To address this issue, one needs to normalize the frequency count of events in the log. Note, however, that often the amount of load itself is a contributing factor to the cause of a bug manifestation. Hence, the best normalization is application specific. This issue has been recently discussed (in the specific context of sensor network debugging) in Dr. Khan's Ph.D. thesis [46].

- *Handling multi-parameter events:* There are also issues with handling event parameters. Since event parameter values may be different, calling each possible combination of parameter values of an event a different name will cause a combinatorial explosion of the alphabet. To address the problem, continuous or fine-grained parameters need to be discretized into a smaller number of ranges. Multi-parameter events need to be converted into sequences of single-parameter events each listing one parameter at a time. Hence, the exponential explosion is reduced to linear growth in the alphabet. Techniques for dealing with event parameter lists were introduced in [48].

An example bug that Dustminer was reported to diagnose using sequence mining is a problem with a multichannel MAC protocol that

occasionally exhibited a lower throughput compared to a single-channel MAC. In this protocol [58], nodes that communicate frequently are clustered together and assigned the same home (radio) channel, whereas nodes that communicate less frequently are clustered into different channels. Hence, communication between nodes in the same cluster is fast. When a node wants to contact another in a different cluster, it needs to switch to that cluster's radio frequency (called its *home frequency*[2]), perform the communication, then return to its own home channel. Since home frequencies change from time to time, as an added robustness mechanism, if it happens that a node cannot find its recipient on what it believes to be the recipient's home frequency, the node starts a scanning procedure, where it scans all channels looking for the recipient, and (having found it) updates its records accordingly. This robustness mechanism ensures that nodes will eventually resume communication, even if updates regarding changes in their home channel are lost.

During experimentation with the protocol, it was noticed that when data rates were low, the multi-channel protocol outperformed a single channel MAC protocol comfortably as it should. However, when the data rates were increased, the protocol performed worse than a single channel MAC. Traces from the two scenarios gave rise to the "good" and "bad" logs, respectively. Diagnosis revealed that the problem lied with the mechanism used by a node to find another in a different cluster. Ironically, the problem was caused by the robustness mechanism described above; namely, a node who failed to communicate with a recipient in a different cluster would time-out and start to scan all channels looking for it. This scanning took a long time, during which the node was not on its home channel. During that time, other nodes *within the same cluster* who wanted to communicate with it would time-out and start scanning as well. Eventually, their predecessors would time-out and start scanning too. Hence, the scanning quickly propagated upstream from node to node, even *within clusters*, until all nodes were scanning in an attempt to find each other, instead of productively communicating. The particular discriminative sequence of events that revealed this problem was described in [49].

3.2 Graph Mining

While sequence mining is a powerful means for identifying chains of events that lead to software problems, some problems are not caused

[2]We use the terms "frequency" and "channel" interchangeably here to refer to a communication channel defined by a different radio frequency.

by a linear chain of events. Rather they are caused by more complex event patterns described more generally by *graphs*. For example, in a sensor data fusion system, if a fusion stage collects data from two children (down the fusion tree) that are incompatible or inconsistent in some way, fusion results may be incorrect. There are many ways such incompatibility or inconsistency may occur. For instance, if in a target tracking algorithm two sensors report distances to *different targets* but give their targets the same ID, a subsequent target triangulation stage may compute an incorrect target location. If two temperature sensors report a -31 and -35 degree temperature, respectively, but use different units, if units are not explicitly stated, a subsequent averaging stage will compute an average that is incorrect in either unit. Here the issue lies in inconsistent assumptions among nodes (e.g., inconsistent assumptions regarding target identity, or inconsistent assumptions regarding used units). Inconsistent assumptions do not mean that one of them is wrong. For example, reporting the average temperature in either unit would have been fine, as long as sensors agreed on the reporting unit. Also, reporting the location of either target would have been fine as long as nodes agreed on the reported target. It is the fusion of data from nodes who disagree on assumptions that is the problem. This is not a problem with the *sequence* of reporting. It is a problem best correlated to the topology of the reporting *tree*, which includes two incompatible children who report to the same parent. Hence, debugging such problems calls for a discriminative pattern mining algorithm that looks for patterns that are more complex than linear sequences, such as trees or more general graphs.

PopMine [81] was the first tool in sensor network literature that used *discriminative graph mining* for diagnosing corner-case bugs. The tool identifies the minimal causal directed acyclic graph (DAG) of events, spanning multiple nodes, that captures a bug-triggering condition. Being based on causal order, a global notion of time is not required in uncovering bug-triggering distributed event patterns. Bug triggering event DAGs are identified by comparing execution graphs from successful runs to those where bug manifestations are observed, and exposing the minimal discriminative event DAGs that may be responsible for the problem.

PopMine significantly extended prior sensor networks debugging tools, based on data mining. Prior work considered simpler bug-triggering conditions such as single events, event sets, or ordered chains of events, as opposed to distributed event graphs.

As is the case with other tools discussed in this chapter, the input to PopMine is "good" and "bad" execution event traces from runs of

different nodes of a distributed system. Each such trace represents a linear event sequence that logs the execution order of events on a single processor in the system. The tool first finds all communication events and matches each sending event to the corresponding receiving event (for each message). The resulting arcs, together with the original event sequences give rise to event graphs, which are then mined for discriminative patterns. The discriminative graph mining algorithm first identifies individual discriminative events, then grows these events recursively into bigger patterns using a set of rules that aim to maximize the information gain (i.e., discriminative ability) of the resulting pattern at each step. A search tree is created whose root is a single discriminative event, and where every node is a discriminative event subgraph. A node's childern represent all ways the (discriminative) event subgraph at the parent node can be extended by one event. Additional rules specify how to grow a pattern to make sure that the space of patterns reached from different search tree branches is not overlapping. For example, a pattern where event `a` is followed by `b` should not be a child of both of the aforementioned individual events. Rather, growth should occur in a specified direction only to eliminate any possible redundancy between different search tree branches.

The tool was used to debug a sensor data aggregation and regression modeling framework that collected measurements from on-board diagnostic interface ports (OBD-II ports) of different vehicles while testing a new navigation service [29]. It was found that, depending on which vehicles were involved in the data collection, some resulting computed models of the data were very poor. Using PopMine, the found bug was attributed to a condition where measurements were collected from two vehicles reporting data in conflicting units. Note that, reporting in either unit consistently would have worked (generating results in that unit), but aggregating conflicting units resulted in an error. The bug triggering condition, in this case, was a graph in which two modules (the two cars with conflicting unit settings) fed a single aggregator. By inspecting the data feeds from the identified cars manually, the issue of conflicting units was recognized.

3.3 Symbolic Pattern Mining

The above "unit mismatch" example, upon closer examination, triggers the observation that discriminative subgraph mining and its predecessors (e.g., discriminative event mining, itemset mining, and sequence mining) suffer from the same inefficiency: namely, they are unable to generalize from individual examples to categories that satisfy a particu-

lar mathematical or logical invariant. For instance, if a large number of cars were involved and many unit incompatibilities were present, rather than returning all combinations of cars that report measurements in conflicting units as discriminative, it would have been good to return a single rule of the form `A` $\rightarrow$ `C , B` $\rightarrow$ `C: A (unit)` $\neq$ `B (unit)`, indicating that the discriminative subgraph features two arcs (between two sources and a common sink) feeding data of different units. The rule could then have support that enumerates all found instances of incompatibility satisfying the above condition. Identifying such symbolic rules could significantly improve the readability of diagnostic results.

An attempt to handle this problem is described in recent literature [47], defining a *symbolic pattern* as one where all or a subset of the absolute values of event attributes within the discriminative pattern (that represent a potential bug triggering condition) are replaced with *symbols* to generalize the pattern. In this case, the logged execution events can include any operations performed at runtime such as message transmission, message reception, and writing to flash storage. Each recorded event can have multiple attributes. The symbolic pattern extraction algorithm first generates discriminative patterns then generalizes them by mining for "relationships" across those patterns. A new scheme is presented for counting the support for individual patterns which greatly enhances the chances of identifying "infrequent" events that are correlated with failure. The resulting patterns are ranked, such that most informative patterns are presented first. The authors demonstrated that the approach returned significantly fewer yet more informative patterns compared previous discriminative mining solutions that had no symbolic generalization capability.

4. Other Sensor Network Debugging Work

There has been a substantial amount of work on automating sensor network debugging, attributed to the inherent difficulty of addressing this problem manually. While the current chapter surveyed approaches based on data mining algorithms, the list of different debugging tools and techniques does not stop there.

Aside from techniques inspired by data mining, work on sensor network debugging started with the development of appropriate *laboratory testbeds*, such as Motelab [94], Kansei [26], and Emstar [32], that facilitate sensor software testing by providing the convenience of a controlled experimental environment. Early work also included tools that improved the visualization of network statistics. One of the first examples in that area is SNMS [88]. It constitutes a sensor network management service

that collects and summarizes different types of data such as packet loss and radio energy consumption to aid humans in uncovering behavior anomalies.

To allow *GDB-like tracing* through distributed sensor network code, sophisticated software tools such as Clairvoyant [98] and Marionette [96] were developed that provide standard debugging support such as breakpoints and watchpoints. Software tracing techniques were also developed [83], where challenges included efficient compression of the execution traces on resource-limited sensor nodes (called *motes*) for later inspection at a base-station. A recently-proposed hardware-assisted tool, called Aveksha [85], introduced an FPGA to interface to the JTAG (debug) port of the embedded processor of a remotely-deployed mote, allowing remote breakpoint, tracepoint, and program-counter tracing functionality, as well as energy profiling.

In a marked departure from techniques that focus on troubleshooting node-level code and data, the concept of *macrodebugging* was recently introduced [82], referring to GDB-like debugging of sensor network *macro-programs*. Macro-programming refers to network-level programming techniques that abstract entire sensor networks in a fashion that allows reasoning about and controlling network behavior as a whole, as opposed to encoding the behavior of individual nodes. Examples of such abstractions include sensor neighborhoods [95], node arrays [53, 34], abstract regions [93], stream feeds [22], and database tables [69]. The macrodebugger [82] allows stepping through the distributed execution of macro-programs, visualize distributed state, and experiment with hypothetical changes to variables. It was implemented on a macro-programming framework, called MacroLab [45].

Prior work also addressed *automatic error detection* in sensor networks based on simple inspection of collected traces (typically integrated with specific sensor network operating systems such as TinyOS [59], SOS [37], Mantis [9], and LiteOS [12]). For example, H-SEND [44] allows TinyOS programmers to specify invariants that must be satisfied at run-time, and inserts the code needed to monitor such invariants and report violations. Similarly, HERMES [55] allows SOS programmers to interpose monitoring code, such as conditional watchpoints, into sensor network programs, as well as to define how certain detected conditions are to be dealt with. NodeMD [56] is designed (and implemented on top of Mantis OS [9]) to detect a category of node-level faults such as stack overflow, deadlock, and livelock. For LiteOS, recent work introduces declarative tracepoints [13], a debugging abstraction that borrows from aspect-oriented computing and allows conditional monitoring and response to selected events. Recent work also considered data errors [35].

Specifically, assuming that sensor data values change monotonically with distance from the sensed event, violations of this monotonicity property can be indicative of incorrect data. This insight was used to detect faulty nodes.

Much work was done on memory safety of TinyOS, including Safe TinyOS [18], Deputy [17], and Neutron [14], which helps nodes survive memory safety violations by definiding recovery units that can be individually restarted, and allowing the programmer to specify precious state that must be kept across restarts whenever possible. Extensions of that work to more general model-checking systems ensued [11, 100, 71], coming close to verifying entire distributed programs written in common sensor network programming languages such as NesC [31].

Finally, outside sensor networks, using machine learning techniques to diagnose failures is not new [10, 3, 63, 27, 64, 61]. Examples include discriminative pattern analysis [27, 66], software behavior graph analysis [63], mining message sequence graphs [57], a Bayesian analysis based approach [62], and control-flow analysis to identify logical errors [64], just to name a few. A recent book describes methodologies and applications of mining software specifications [67].

5. Future Challenges

While a significant amount of work was done on sensor network debugging to date, several interesting opportunities remain. In general, there is room for improving the scalability of current approaches. Better data mining tools can be brought to bear to address more subtle bug patterns. For example, graphs that describe bug triggers should be weighted to reflect the fact that certain events (vertices in the graph) cause *multiple instances* of other events (other vertices), which can be compactly represented as an edge weight. Hence, discriminative mining algorithms are needed for weighted graphs.

The issue of concurrent bugs, where multiple different causes give rise to common symptoms is another challenge that decreases efficacy of current discriminative techniques when no single cause has enough support to definitively account for the observed problem. Diagnosing rare (anomalous) events is also troublesome, since very little observations may be available about the anomaly, while a great predominance of data is obtained during normal operation. Asynchrony (and lack of global time) in distributed systems further complicates reconstruction of exact event patterns that cause an anomaly. Since both causes and symptoms are correlated with the occurrence of problems, disambiguating the two remains important. It requires the ability to reason about causal links

between different events, which goes beyond observing mere correlations or association rules.

Besides the aforementioned extensions, significant opportunities are present for landmark results that define new areas. In particular, current work broadly adopts one of two distinct philosophies; either (i) eliminating errors by design, or (ii) troubleshooting them as they occur. Present literature on sensor network debugging is fragmented along the above boundary into *pre-run-time* and *run-time* solutions. Pre-run-time solutions are motivated by the potentially great financial and safety cost of run-time errors, whereas run-time solutions are sought due to the scalability challenges and high cost of reliably proving correctness in advance. For a great category of embedded and networked sensing systems, both solutions are of value, as they complement each other's limitations. Hence, an interesting area of investigation lies at the intersection between the two. It would be of great interest to understand, for example, how run-time data mining solutions can help build increasingly more reliable and complete models of subsystems, suitable for formal reasoning about correctness when these subsystems are integrated into other systems. It is also interesting to understand how an incomplete model of a system (e.g., a detailed model of only some of the components) can be integrated with data mining solutions to significantly improve the scalability and accuracy of root-cause diagnosis.

A different open area lies in the topic of learning from experience. When human experts troubleshoot systems, they come with significant background and experience inherited from other systems that helps them identify new problems quickly. How to develop a system representation model then, such that troubleshooting tools can learn from their own experience and decide which parts of a learned model remain applicable in a new context? How to exploit such past experience to substantially improve the scalability and diagnostic accuracy of debugging tools? While significant work has been done that demonstrates instances of such transfer learning, general solutions that apply experiences across broad categories of networked sensing installations remain to be found. Note that, this transfer learning problem, in the context of networked sensing, is more challenging than its counterpart in general distributed computing. Many different distributed computing systems are built on the same operating system abstractions, such as threads, processes, memory, system calls, and synchronization primitives, making it easier to transfer knowledge across troubleshooting experiences. For example, the symptoms of a deadlock are similar across different software systems. In contrast, many embedded system installations are unique. They use different sensors and actuators in different types of physical environments, giving

rise to unique interactions and failure modalities that are harder to learn from and generalize.

Since a user or a system analyst may have good knowledge on where to find bugs or how to control the search space in the debugging process, it could also be interesting to investigate how to integrate user-specified constraints into the debugging process. Constraint-based mining of association or correlation rules has been studied in data mining literature. However, there is still lack of study on how to perform constraint-based bug analysis in sensor networks, that is, how to use user-specified constraints to influence the process of search and analysis of potential bugs. This is an important challenge in integrating data mining with sensor network debugging.

More generally, the interaction between sensor network debugging tools and users is an important topic. Rather than aiming to replace the human entirely, tools should leverage and augment human capabilities. Interactions with a user could, for example, take the form of a progressive drill-down pattern, where the cause of a problem is incrementally unveiled via a series of progressive refinement steps. This pattern has significant scalability implications. Rather than searching a very large problem space exhaustively at the outset, the tool would cover it at a high level, then prune significant portions before drilling down into a subspace at the next level of detail. Hence, an interesting challenge is to design tools that use minimum resources by exloiting some form of progressive drill-down. Solving the above challenges can significantly impact the cost of development and ownership of networked sensing systems, which may in turn increase the scope of potential applications, as well as eventually the reliability of deployed sensing systems.

References

[1] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, and A. Wood. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, ICDCS '04, pages 582–589, Washington, DC, USA, 2004. IEEE Computer Society.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, 1994.

[3] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed sys-

tems of black boxes. In *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP'03)*, pages 74–89, 2003. Bolton Landing, NY, USA.

[4] A. Al-Nayeem, L. Sha, D. Cofer, and S. Miller. Pattern-based composition and analysis of virtually synchronized real-time distributed systems. In *International Conference on Cyber-physical Systems*, April 2012.

[5] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[6] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha. The system-level simplex architecture for improved real-time embedded system safety. In *Proceedings of the 2009 15th IEEE Symposium on Real-Time and Embedded Technology and Applications*, RTAS '09, pages 99–107, Washington, DC, USA, 2009. IEEE Computer Society.

[7] S. Bak, A. Greer, and S. Mitra. Hybrid cyberphysical system verification with simplex using discrete abstractions. In *IEEE Technology and Applications Symposium*, April 1996.

[8] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo. Sandboxing controllers for cyber-physical systems. In *International Conference on Cyber-physical Systems*, April 2011.

[9] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 10(4):563–579, August 2005.

[10] P. Bodik, G. Friedman, L. Biewald, H. Levine, G. Candea, K. Patel, G. Tolle, J. Hui, A. Fox, M. I. Jordan, and D. Patterson. Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization. In *Proceedings of the 2nd International Conference on Autonomic Computing(ICAC'05)*, 2005.

[11] D. Bucur and M. Z. Kwiatkowska. Software verification for tinyos. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 400–401, New York, NY, USA, 2010. ACM.

[12] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He. The liteos operating system: Towards unix-like abstractions for wireless sensor networks. In *Proceedings of the Seventh International Conference on Information Processing in Sensor Networks (IPSN'08)*, April 2008.

[13] Q. Cao, T. Abdelzaher, J. Stankovic, K. Whitehouse, and L. Luo. Declarative tracepoints: A programmable and application independent debugging system for wireless sensor networks. In *In Proc. of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 85–98, Raleigh, NC, November 2008.

[14] Y. Chen, O. Gnawali, M. Kazandjieva, P. Levis, and J. Regehr. Surviving sensor network software faults. In *In Proc. of the 22nd ACM Symposium on Operating Systems Principles (SOSP'09)*, pages 235–246, Big Sky, MT, October 2009.

[15] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 169–178, Washington, DC, USA, 2008. IEEE Computer Society.

[16] Y. Chevalier, L. Compagna, J. Cuellar, P. Drielsma, J. Mantovani, S. odersheim, and L. Vigneron. A high level protocol specification language for industrial securitysensitive protocols. In *Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS)*, pages 193–205, 2004.

[17] J. Condit, M. Harren, Z. Anderson, D. Gay, and G. C. Necula. Dependent types for low-level programming. In *In Proc. of the 16th European Symp. on Programming (ESOP)*, Braga, Portugal, March 2007.

[18] N. Cooprider, W. Archer, E. Eide, D. Gay, and J. Regehr. Efficient memory safety for tinyos. In *In Proc. of the 5th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 205–218, Sydney, Australia, November 2007.

[19] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar. The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium*, RTSS '07, pages 400–412, Washington, DC, USA, 2007. IEEE Computer Society.

[20] T. L. Crenshaw, C. L. Robinson, H. Ding, P. R. Kumar, and L. Sha. A pattern for adaptive behavior in safety-critical, real-time middleware. In *IEEE RTSS*, December 2006.

[21] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, 2003.

[22] R. Dickerson, J. Lu, J. Lu, and K. Whitehouse. Stream feeds: an abstraction for the world wide sensor web. In *Proceedings of the 1st*

international conference on The internet of things, IOT'08, pages 360–375, Berlin, Heidelberg, 2008. Springer-Verlag.

[23] B. Ding, D. Lo, J. Han, and S.-C. Khoo. Efficient mining of closed repetitive gapped subsequences from a sequence database. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 1024–1035, Washington, DC, USA, 2009. IEEE Computer Society.

[24] J. S. Dong, J. Sun, J. Sun, K. Taguchi, and X. Zhang. Specifying and verifying sensor networks: An experiment of formal methods. In *Proceedings of the 10th International Conference on Formal Methods and Software Engineering*, ICFEM '08, pages 318–337, Berlin, Heidelberg, 2008. Springer-Verlag.

[25] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, LCN '04, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[26] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko. Kansei: A testbed for sensing at scale. In *Proceedings of the 4th Symposium on Information Processing in Sensor Networks (IPSN/SPOTS track)*, 2006.

[27] G. D. Fatta, S. Leue, and E. Stegantova. Discriminative pattern mining in software fault detection. In *Proceedings of the 3rd international workshop on Software quality assurance (SOQUA '06)*, pages 62–69, 2006.

[28] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, pages 144–151, 1998.

[29] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. Greengps: a participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 151–164, New York, NY, USA, 2010. ACM.

[30] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining very large databases. *COMPUTER*, 32:38–45, 1999.

[31] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, PLDI '03, pages 1–11, New York, NY, USA, 2003. ACM.

[32] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC'04)*, pages 24–24, Boston, MA, 2004.

[33] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM.

[34] R. Gummadi, N. Kothari, R. Govindan, and T. Millstein. Kairos: a macro-programming system for wireless sensor networks. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, SOSP '05, pages 1–2, New York, NY, USA, 2005. ACM.

[35] S. Guo, Z. Zhong, and T. He. Find: faulty node detection for wireless sensor networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 253–266, New York, NY, USA, 2009. ACM.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[37] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, pages 163–176, New York, NY, USA, 2005. ACM.

[38] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, Aug. 2007.

[39] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann, 2011.

[40] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM.

[41] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[42] Y. Hanna, H. Rajan, and W. Zhang. Slede: a domain-specific verification framework for sensor network security protocol implementations. In *Proceedings of the first ACM conference on Wireless*

network security, WiSec '08, pages 109–118, New York, NY, USA, 2008. ACM.

[43] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.)*. Springer-Verlag, 2009.

[44] D. Herbert, V. Sundaram, Y.-H. Lu, S. Bagchi, and Z. Li. Adaptive correctness monitoring for wireless sensor networks using hierarchical distributed run-time invariant checking. *ACM Trans. Auton. Adapt. Syst.*, 2(3), Sept. 2007.

[45] T. W. Hnat, T. I. Sookoor, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrolab: a vector-based macroprogramming framework for cyber-physical systems. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 225–238, New York, NY, USA, 2008. ACM.

[46] M. Khan. *Troubleshooting Interactive Complexity Bugs*. PhD thesis, University of Illinois at Urbana Champaign, 2011.

[47] M. M. Khan, T. Abdelzaher, J. Han, and H. Ahmadi. Finding symbolic bug patterns in sensor networks. In *Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '09, pages 131–144, Berlin, Heidelberg, 2009. Springer-Verlag.

[48] M. M. H. Khan, T. Abdelzaher, and K. K. Gupta. Towards diagnostic simulation in sensor networks. In *Proceedings of International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2008. Greece.

[49] M. M. H. Khan, H. K. Le, H. Ahmadi, T. F. Abdelzaher, and J. Han. Dustminer: Troubleshooting interactive complexity bugs in sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008. Raleigh, NC, USA.

[50] M. M. H. Khan, H. K. Le, M. LeMay, P. Moinzadeh, L. Wang, Y. Yang, D. K. Noh, T. Abdelzaher, C. A. Gunter, J. Han, and X. Jin. Diagnostic powertracing for sensor node failure analysis. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 117–128, New York, NY, USA, 2010. ACM.

[51] M. M. H. Khan, L. Luo, C. Huang, and T. Abdelzaher. Snts: Sensor network troubleshooting suite. In *Proceedings of International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2007. Santa Fe, New Mexico, USA.

[52] H. Kim, S. Kim, T. Weninger, J. Han, and T. Abdelzaher. Ndpmine: efficiently mining discriminative numerical features for pattern-based classification. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part II*, ECML PKDD'10, pages 35–50, Berlin, Heidelberg, 2010. Springer-Verlag.

[53] N. Kothari, R. Gummadi, T. Millstein, and R. Govindan. Reliable and efficient programming abstractions for wireless sensor networks. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '07, pages 200–210, New York, NY, USA, 2007. ACM.

[54] N. Kothari, T. Millstein, and R. Govindan. Deriving state machines from tinyos programs using symbolic execution. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 271–282, Washington, DC, USA, 2008. IEEE Computer Society.

[55] N. Kothari, K. Nagaraja, V. Raghunathan, F. Sultan, and S. Chakradhar. Hermes: A software architecture for visibility and control in wireless sensor network deployments. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 395–406, Washington, DC, USA, 2008. IEEE Computer Society.

[56] V. Krunic, E. Trumpler, and R. Han. Nodemd: Diagnosing node-level faults in remote wireless sensor systems. In *In Proc. of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 43–56, San Juan, Puerto Rico, June 2007.

[57] S. Kuma, S.-C. Khoo, A. Roychoudhury, and D. Lo. Mining message sequence graphs. In *In Proc. Int. Conf. on Software Eng. (ICSE '11)*, Honolulu, HI, May 2011.

[58] H. K. Le, D. Henriksson, and T. Abdelzaher. A practical multi-channel media access control protocol for wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 70–81, Washington, DC, USA, 2008. IEEE Computer Society.

[59] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks ambient intelligence. *Ambient Intelligence*, pages 115–148, 2005.

[60] P. Li and J. Regehr. T-check: Bug finding for sensor networks. In *In Proc. of the 9th ACM/IEEE International Conference on*

Information Processing in Sensor Networks (IPSN), pages 174–185, Stockholm, Sweden, April 2010.

[61] C. Liu, L. Fei, X. Yan, J. Han, and S. P. Midkiff. Statistical debugging: A hypothesis testing-based approach. *IEEE Transactions on Software Engineering*, 32:831–848, 2006.

[62] C. Liu, Z. Lian, and J. Han. How bayesians debug. In *Proceedings of the Sixth International Conference on Data Mining (ICDM'06)*, pages 382–393, December 2006.

[63] C. Liu, X. Yan, L. Fei, J. Han, and S. P. Midkiff. Sober: statistical model-based bug localization. In *Proceedings of the 13th ACM SIGSOFT international symposium on Foundations of software engineering (FSE-13)*, 2005. Lisbon, Portugal.

[64] C. Liu, X. Yan, and J. Han. Mining control flow abnormality for logic error isolation. In *Proceedings of 2006 SIAM International Conference on Data Mining (SDM'06)*, Bethesda, MD, April 2006.

[65] K. Liu, M. Li, Y. Liu, M. Li, Z. Guo, and F. Hong. Passive diagnosis for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 113–126, New York, NY, USA, 2008. ACM.

[66] D. Lo, H. Cheng, J. Han, S. Khoo, and C. Sun. Classification of software behaviors for failure detection: A discriminative pattern mining approach. In *In Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '09)*, Paris, France, June 2009.

[67] D. Lo, S.-C. Khoo, J. Han, and C. Liu. *Mining Software Specifications: Methodologies and Applications*. Chapman and Hall, 2011.

[68] J. MacLennan, Z. Tang, and B. Crivant. *Data Mining with SQL Server 2008*. John Wiley & Sons, 2008.

[69] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, Mar. 2005.

[70] Z. Markov and I. Russell. An introduction to the weka data mining system. In *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, ITICSE '06, pages 367–368, New York, NY, USA, 2006. ACM.

[71] L. Mottola, T. Voigt, F. Österlind, J. Eriksson, L. Baresi, and C. Ghezzi. Anquiro: enabling efficient static verification of sensor network software. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Sensor Network Applications*, SESENA '10, pages 32–37, New York, NY, USA, 2010. ACM.

[72] N. T. M. Nguyen and M. L. Soffa. Program representations for testing wireless sensor network applications. In *Workshop on Domain specific approaches to software test automation: in conjunction with the 6th ESEC/FSE joint meeting*, DOSTA '07, pages 20–26, New York, NY, USA, 2007. ACM.

[73] P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of real-time maude. *Higher Order Symbol. Comput.*, 20(1-2):161–196, June 2007.

[74] P. C. Olveczky and S. Thorvaldsen. Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in real-time maude. *Theor. Comput. Sci.*, 410(2-3):254–280, Feb. 2009.

[75] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 16(11):1424–1440, 2004.

[76] C. Perrow. *Normal Accidents: Living With High-Risk Technologies*. Princeton University Press, 1984.

[77] P. A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, 2000.

[78] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys'05)*, pages 255–267, 2005.

[79] R. Sasnauskas, O. Landsiedel, M. Alizai, C. Weise, S. Kowalewski, and K. Wehrle. Kleenet: Discovering insidious interaction bugs in wireless sensor networks before deployment. In *In Proc. of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 186–196, Stockholm, Sweden, April 2010.

[80] R. Sasnauskas, J. A. B. Link, M. H. Alizai, and K. Wehrle. Kleenet: automatic bug hunting in sensor network applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 425–426, New York, NY, USA, 2008. ACM.

[81] E. Seo, M. M. H. Khan, P. Mohapatra, J. Han, and T. Abdelzaher. Exposing complex bug-triggering conditions in distributed systems via graph mining. In *Proceedings of the 2011 International Confer-*

ence on Parallel Processing, ICPP '11, pages 186–195, Washington, DC, USA, 2011. IEEE Computer Society.

[82] T. Sookoor, T. Hnat, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrodebugging: global views of distributed program execution. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 141–154, New York, NY, USA, 2009. ACM.

[83] V. Sundaram, P. Eugster, and X. Zhang. Efficient diagnostic tracing for wireless sensor networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 169–182, New York, NY, USA, 2010. ACM.

[84] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.

[85] M. Tancreti, M. S. Hossain, S. Bagchi, and V. Raghunathan. Aveksha: a hardware-software approach for non-intrusive tracing and profiling of wireless embedded systems. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 288–301, New York, NY, USA, 2011. ACM.

[86] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[87] A. Tiwari, R. Gupta, and D. Agrawal. A survey on frequent pattern mining: Current status and challenging issues. *Information Technology Journal*, pages 1278–1293, 2010.

[88] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN'05)*, pages 121–132, Istanbul, Turkey, February 2005.

[89] T. Uno, T. Asai, Y. Uchida, and H. Arimura. Lcm: An efficient algorithm for enumerating frequent closed item sets. In *In Proceedings of Workshop on Frequent itemset Mining Implementations (FIMI, 03*, 2003.

[90] V. N. Vapnik. *The nature of statistical learning theory*. Springer, 2000.

[91] P. Völgyesi, M. Maróti, S. Dóra, E. Osses, and A. Lédeczi. Software composition and verification for sensor networks. *Sci. Comput. Program.*, 56(1-2):191–210, Apr. 2005.

[92] S. M. Weiss and N. Indurkhya. *Predictive Data Mining*. Morgan Kaufmann, 1998.

[93] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 3–3, Berkeley, CA, USA, 2004. USENIX Association.

[94] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, pages 483–488, April 2005.

[95] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: a neighborhood abstraction for sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 99–110, New York, NY, USA, 2004. ACM.

[96] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler. Marionette: Using rpc for interactive development and debugging of wireless embedded networks. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks: Special Track on Sensor Platform, Tools, and Design Methods for Network Embedded Systems (IPSN/SPOTS)*, pages 416–423, Nashville, TN, April 2006.

[97] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.). Morgan Kaufmann, 2005.

[98] J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: a comprehensive source-level debugger for wireless sensor networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys'07)*, pages 189–203, 2007.

[99] M. J. Zaki and C. jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SIAM International Conference on Data Mining(SDM)*, pages 457–473, 2002.

[100] M. Zheng, J. Sun, Y. Liu, J. S. Dong, and Y. Gu. Towards a model checker for nesc and wireless sensor networks. In *Proceedings of the 13th international conference on Formal methods and software engineering*, ICFEM'11, pages 372–387, Berlin, Heidelberg, 2011. Springer-Verlag.

[101] Y. Zhou, X. Chen, M. Lyu, and J. Liu. Sentomist: Unveiling transient sensor network bugs via symptom mining. In *In Proc. 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 784–794, Genoa, Italy, June 2010.

Chapter 14

MINING OF SENSOR DATA IN HEALTHCARE: A SURVEY

Daby Sow
IBM T. J. Watson Research Center
Hawthorne, NY
sowdaby@us.ibm.com

Deepak S. Turaga
IBM T. J. Watson Research Center
Hawthorne, NY
turaga@us.ibm.com

Michael Schmidt
Columbia University Medical Center
Neurological ICU, New York, NY
mjs2134@mail.cumc.columbia.edu

Abstract Historically, healthcare has been mainly provided in a reactive manner that limits its usefulness. With progress in sensor technologies, the instrumentation of the world has offered unique opportunities to better observe patients physiological signals in order to provide healthcare in a more proactive manner. To reach this goal, it is essential to be able to analyze patient data and turn it into actionable information using data mining. This chapter surveys existing applications of sensor data mining technologies in healthcare. It starts with a description of healthcare data mining challenges before presenting an overview of applications of data mining in both clinical and non clinical settings.

Keywords: healthcare, data mining, physiological sensors,

1. Introduction

Healthcare includes "efforts made to maintain or restore health, especially by trained and licensed professionals" [1]. These efforts are performed by various entities within a large ecosystem composed of patients, physicians, payers, health providers, pharmaceutical companies and more recently, IT companies. Medical informatics [2] is the science that deals with health information, its structure, acquisition, and use. A fundamental goal [3] of medical informatics is the improvement of healthcare by acquiring and transmitting knowledge for applications in a broad range of settings, across computational platforms, and in a timely fashion.

Reaching this goal can have far-reaching consequences on our society. Historically, healthcare has been provided in a reactive manner that limits its usefulness. A major issue with this is the inability to detect early or predict that a patient may be prone to develop complications associated with chronic diseases like cancer or diabetes. Even in an intensive care environment, care is often provided in response to the adverse events typically detected after the emergence of clinical symptoms, or after the interpretation of a lab test. Quite often, reacting after the detection of such events reduces the ability of physicians to drive patient trajectories towards good outcomes. As a result, there is an increasing push to transform medical care delivery from reactive to proactive.

This transformation necessitates better monitoring and understanding of the patients. Medical institutions and healthcare providers are collecting large amounts of data on their patients, and organizing this data into Electronic Medical Records (EMR) and Patient Health Records (PHR). With the advances in sensor technologies, several new data sources, providing insights on patients, are emerging. For instance, Bluetooth enabled scales, blood pressure cuffs, heart rate monitors or even portable electrocardiogram monitors are now available off the shelves for the collection of important vitals that can be interpreted for early diagnosis. Using these advances in sensor technologies, several remote health monitoring solutions for chronic disease management, and wellness management have been proposed [4].

While this rapid growth in healthcare sensor data offers significant promise to impact care delivery, it also introduces a data overload problem, for both systems and stakeholders that need to consume this data. It is, therefore necessary to complement such sensing capabilities with data mining and analytical capabilities to transform the large volumes of collected data into meaningful intelligence. In this chapter, we survey the application of sensor data mining technologies in medical in-

formatics. We divide this application space into two parts: clinical and non-clinical applications. Clinical applications are essentially clinical decision support applications for both in- and out-patient scenarios. Non clinical applications include wellness management, activity monitoring, the use of smart environments (e.g., smart home scenarios) and reality mining. We provide a detailed survey of the sensors, systems, analytic techniques, and applications and challenges in these different areas, in this chapter.

The rest of this chapter is organized as follow. In Section 2, we present research challenges associated with the mining of sensor data in medical informatics. In Section 3.1, we review sensor mining applications and systems in clinical healthcare settings, while in Section 4 we describe several applications in non-clinical settings. We conclude in Section 5.

2. Mining Sensor Data in Medical Informatics: Scope and Challenges

Sensors measure physical attributes of the world and produce signals, i.e. time series consisting of ordered sequences of pairs (timestamps,data elements). For example, in intensive care, respiration rates are estimated from measurements of the chest impedance of the patient. The resulting time series signals are consumed either by a human or by other sensors and computing systems. For instance, the output of the chest impedance sensor may be consumed by an apnea detection sensor to produce a signal measuring apnea episodes. The data elements produced by sensors range from simple scalar numerical or categorical values, to complex data structures. Examples of simple data elements include measures such as hourly average of temperature in a given geographical location, output by a temperature sensor. Examples of more complex data elements include summaries of vital signs and alerts measured by a patient monitor sensor in a medical institution. In this chapter, we focus on sensing challenges for medical informatics applications.

2.1 Taxonomy of Sensors used in Medical Informatics

As shown in Figure 14.1, we categorize sensors in medical informatics as follows:

- Physiological sensors: These sensors measure patient vital signs or physiological statistics. They were first used to measure vitals on astronauts before appearing in medical institutions, at the bedside in the 1960s. Today, physiological sensors are also available out-

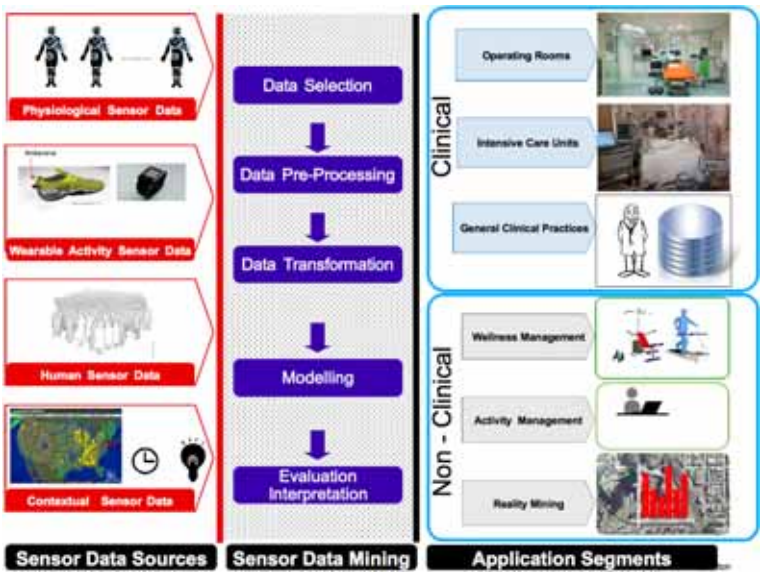


Figure 14.1. The sensor data mining process.

side medical institutions, even on pervasive devices (e.g., iPhone heart rate monitor applications that make use of smart phone cameras [5]).

- Wearable activity sensors: These sensors measure attributes of gross user activity, different from narrowly focused vital sign sensors. Good examples are accelerometers used for gait monitoring. Shoe manufacturers like Nike have enabled many of their running shoes with sensors capable of tracking walking or jogging activities [6]. Most smart phones are also equipped with accelerometers and several wellness management applications leverage these sensors.

- Human sensors: Humans play an integral role in the sensing process. For instance, physicians introduce important events that relate to the patient health status during examinations. Lab technicians follow rigorous processes to provide blood content information. Self-reporting (i.e. patients monitoring their health parameters) is also used in the management of chronic illnesses like diabetes. More recently, with the emergence of social media and pervasive computing, people use mechanisms like web-searches and Twitter to generate reports on important health related events.

- Contextual sensors: These sensors are embedded in the environment around the user to measure different contextual properties.

Examples include motion detection sensors, audio and video sensors, temperature sensors, weather sensors, etc.

2.2 Challenges in Mining Medical Informatics Sensor Data

As with standard data mining procedures [7] [8], healthcare mining is typically performed in five stages:

- *(I) Data Acquisition*: This includes operations involved in collecting data from external sensor data sources.
- *(II) Data Pre-processing*: This includes operations applied to the data to prepare it for further analysis. Typical pre-processing operations include data cleaning to filter out noisy data elements, data interpolation to cope with missing values, data normalization to cope with heterogeneous sources, temporal alignment, and data formatting.
- *(III) Data Transformation*: The includes operations for representing the data appropriately and selecting specific features from this representation. This stage is often called feature extraction and selection.
- *(IV) Modeling*: This stage, also called *mining* applies knowledge discovery algorithms to identify patterns in the data. Modeling problems can be classified into six broad categories: (1) anomaly detection to identify statistically deviant data, (2) association rules to find dependencies and correlations in the data, (3) clustering models to group data elements according to various notions of similarity, (4) classification models to group data elements into predefined classes, (5) regression models to fit mathematical functions to data and (6) summarization models to summarize or compress data into interesting pieces of information.
- *(V) Evaluation*: This stage includes operations for evaluation and interpretation of the results of the modeling process.

There are several analytical challenges associated with each of these stages – specific to healthcare mining – that are listed in Table 2.2.

We present these analytical challenges in more detail in the rest of this section.

2.2.1 Acquisition Challenges. Despite several standardization efforts, medical sensor manufacturers tend to design proprietary

Table 14.1. Sensor data mining challenges at each stage of the data mining process

(I) Acquisition	*(II) Pre-processing*
lack of data standards	data formatting
lack of data protocols	data normalization
data privacy	data synchronization
(III) Transformation	*(IV) Modeling*
physiological feature extraction	sequential mining
feature time scales	distributed mining
unstructured data	privacy preserving modeling
	obtaining ground truth
	exploration-exploitation trade-offs
(V) Evaluation and Interpretation	
Model expressiveness	
Process and data provenance	

data models and protocols to externalize sensed signals. In healthcare, standard bodies like HL7 [9] and the Continua Health Alliance [10] address data modeling issues while several IEEE standard protocols address device interoperability issues [11]. However, there is a lack of incentives for sensor data manufacturers to adhere to these standards. With this lack of adherence to standards, mining medical sensor data across multiple data sources involves several non trivial engineering challenges, and the design of custom solutions specific to each sensor data mining application.

Another key challenge in the acquisition process is related to the protection of user privacy. In United States, the Health Insurance Portability and Accountability Act (HIPAA) defines regulations on accesses to health data. By law, data mining applications that leverage this data must comply with these regulations. Data de-identification and de-anonymization techniques are often required to comply with HIPAA. Privacy preserving data mining techniques [12], [13] may also be used to extract information form sensor data while preserving the anonymity of the data.

2.2.2 Pre-processing Challenges. Data in the real world is inherently noisy. The pre-processing stage needs to address this problem

with sophisticated data filtering and interpolation techniques to remove and correct, when possible, data anomalies. The pre-processing stage is also impacted by the lack of standard adoption by medical sensor manufacturers. Indeed, data generated in different formats needs to be syntactically aligned before any analysis can take place. Furthermore, a semantic normalization is often required to cope with differences in the sensing process. As an illustration, a daily reported heart rate measure may correspond to a daily average heart rate in some cases, while in other cases it may represent a heart rate average measured every morning when the subject wakes up. Comparing these values in a data mining application can yield incorrect conclusions, especially if they are not semantically distinguished.

Another key pre-processing challenge involves data synchronization. Sensors report data with timestamps based on their internal clocks. Given that clocks across sensors are often not synchronized, aligning the data across sensors can be quite challenging. In addition, sensors may report data at different rates. Hence, assumptions and alignment strategies need to be carefully designed.

2.2.3 Transformation Challenges. Feature extraction is often the most complex stage of the data mining process. The transformation of sensor data into spaces where good features can be extracted requires a deep understanding of the problem at hand and needs to be driven by domain experts. In medical informatics, this transformation requires expertise on the physiology of the body. Despite immense progress in medicine and in our understanding of the human body, there is still much to learn about all the data that we can sense today. For instance, in neurological intensive care environments, neuro-intensivists collect and interpret electroencephalograms signals that represent the brain activity of their patients. These signals are extremely noisy and not fully understood [14], yet they can be used to diagnose several conditions (e.g., the onset of diverse forms of seizures). Extracting features from EEG signals is often restricted to spectral analysis techniques defined by domain experts.

In addition to signals that are not well understood, human sensing adds different types of unstructured data that needs to be effectively integrated. This includes textual reports from examinations (by physicians or nurses) that also need to be transformed into relevant features, and aligned with the rest of the physiological measurements. These inputs are important to the data mining process as they provide expert data, personalized to the patients. However these inputs can be biased by physician experiences, or other diagnosis and prognosis techniques they

use [15]. Capturing some of these aspects during the mining process is extremely challenging.

2.2.4 Modeling Challenges. There are several challenges that need to be overcome in the modeling stage of the data mining process for medical sensor data. First of all, the time series nature of the data often requires the application of sequential mining algorithms that are often more complex than conventional machine learning techniques (e.g., standard supervised and unsupervised learning approaches). Non-stationarities in time series data necessitate the use of modeling techniques that can capture the dynamic nature of the state of the underlying processes that generate the data. Known techniques for such problems, including discrete state estimation approaches (e.g. dynamic bayesian networks and hidden Markov models) and continuous state estimation approaches (e.g. Kalman Filters or recurrent neural networks) have been used only in limited settings.

Another challenge arises due to the inherent distributed nature of these applications. In many cases, communication and computational costs, as well as sharing restrictions for patient privacy prevent the aggregation of the data in a central repository. As a result, the modeling stage needs to use complex distributed mining algorithms. In remote settings, there is limited control on the data acquisition at the sensor. Sensors may be disconnected for privacy reasons or for resource management reasons (e.g., power constraints), thereby affecting the data available for analysis. Modeling in these conditions may also require the distribution of analytic approaches between the central repository and the sensors. Optimizing the modeling process becomes a challenging distributed data mining problem that has received only limited attention in the data mining community.

Modeling in healthcare mining is also hindered by the ability to obtain ground truth on the data. Labels are often imprecise and noisy in the medical setting. For instance, a supervised learning approach for the early detection of a chronic disease requires well-labeled training data. However, domain experts do not always know exactly when a disease has started to manifest itself in a body, and can only approximate this time. Additionally, there are instances of misdiagnosis that can lead to incorrect or noisy labels that can degrade the quality of any predictive models.

In clinical settings, physicians do not have the luxury of being able to try different treatment options on their patients for exploration purposes. As a result, historical data sets used in the mining process tend to be quite sparse and include natural biases driven by the way care was

delivered to the patient. Standard approaches are not well-equipped to cope with this bias in the data, especially as it is hard to quantify precisely. Furthermore, most studies in medical informatics are retrospective. Well-done prospective studies are hard to do, and are often done on small populations, limiting the statistical significance of any derived results.

2.2.5 Evaluation and Interpretation Challenges. Data mining results consist of models and predictions that need to be interpreted by domain experts. Many modeling techniques produce models that are not easily interpretable. For example, the weights of a neural network may be difficult to grasp for a domain expert. But for such a model to be adopted for clinical use, it needs to be validated with existing medical knowledge. It becomes imperative to track provenance metadata describing the process used to derive any results from data mining to help domain expert interpret these results. Furthermore, the provenance of the data sets, and analysis decisions used during the modeling are also required by the experts to evaluate the validity of the results. This imposes several additional requirements on the selected models and analysis.

2.2.6 Generic Systems Challenges. Beyond analytical challenges, sensor data mining also comes with a set of systems challenges – that apply to medical informatics applications. The mining of sensor data typically requires more than conventional data management (database or data warehousing) technologies for the following reasons:

- The temporal aspect of the data produced by sensors sometimes generate large amounts of data that can overwhelm a relational database system. For example, a large population monitoring solution requiring the real-time analysis of physiological readings, activity sensor readings and social media interactions, cannot be supported with relational database technologies alone.

- Sensor mining applications often have real-time requirements. A conventional store and analyze paradigm with the use of relational database technologies may not be appropriate for such time sensitive applications.

- The unstructured nature of some of the data produced by sensors coupled with the real-time requirements imposes requirements on the programming and analysis models used by developers of sensor data mining applications.

Hence, sensor mining in healthcare requires the use of emerging stream processing system technology in conjunction with database and data warehousing technologies. Stream processing systems are designed to cope with large amounts of real-time data, and their programming models are geared towards the analysis of structured and unstructured sensor data. They are also time sensitive and analyze data within small latency bounds. Figure 14.2 presents an extended architecture for sensor data mining that illustrates this integration. The rationale behind this architecture is to use a stream processing system for the real-time analysis of sensor data, including the pre-processing and transformation stages of the analytical data mining process. The sensor data acquisition is performed by a layer of software that interfaces with sensors and feeds into the stream processing system. The results of the transformation stage may be persisted in a data warehouse for offline modeling with machine learning techniques. The resulting models may be interpreted by analysts and redeployed on the stream processing platform for real-time scoring. In some cases, online learning algorithms may be implemented on the stream processing system. This integration of stream processing with data warehousing technologies creates a powerful architecture that addresses the system challenges outlined above.

3. Sensor Data Mining Applications

As illustrated in Figure 14.1, applications of sensor data mining technologies in healthcare can be classified in two major groups: clinical and non-clinical applications. In the former group, the main data sources subjected to the mining process are direct observations of patient physiological states. In these cases, data mining is often applied to these data sources to build patient models for diagnosis and prognosis. Clinical applications are typically found inside medical institutions. In contrast, non-clinical applications have a broader scope and do not limit themselves to the mining of patient physiological data. Such applications may be found both inside and outside of medical institutions. In the rest of this chapter, we survey the literature in both of these classes.

3.1 Clinical Healthcare Applications

Systems supporting clinical applications of data mining technologies in healthcare are often called Clinical Decision Support Systems (CDSS). CDSSs have been used in both in-patient and out-patient scenarios[1].

[1] In-patient scenarios refer to scenarios for patients that are hospitalized for more than 24 hours. Out-patient scenarios refer to the rest of the clinical use-cases.

CDSSs provide medical practitioners with knowledge and patient-specific information, intelligently filtered and presented at appropriate times, to improve the delivery of care [20]. CDSSs differ from clinical guidelines in that they are more data driven and require access to patient specific data to help physicians in their decision making process. In theory, such systems promise to offer customized and personalized decision support. While several CDSSs are being reported in the literature, few of them make full use of sensor data to assist in the decision making. In this section, we focus on those clinical applications that do use sensor data extensively to aid physicians in their decision making process. We survey applications in intensive care, operating rooms and in general clinical settings.

3.1.1 Intensive Care Data Mining. Today, critically ill patients are often attached to large numbers of body sensors connected to sophisticated monitoring devices producing large volumes of physiological data. Intensive care units are good examples of such data- rich environments where multiple streams of continuous data are typically produced, on a per patient basis. These data streams originate from medical devices that include electrocardiogram, pulse oximetry, electroencephalogram, and ventilators, resulting in several kilobits of data each second. While these monitoring systems aim at improving patient care and staff productivity, they clearly have introduced a data explosion problem. In fact, the vast majority of data collected by these monitoring systems in Intensive Care Units (ICUs) is transient. In talking with medical professionals, we learned that the typical practice in ICUs is for a nurse to eyeball representative readings and record summaries of these readings in the patient record once every 30-60 minutes. The rest of the data remains on the device for 72-96 hours (depending on the memory capacity of the device) before it times out and is lost forever. Hospitals are simply not equipped with the right tools to cope with most of the data collected on their patients, prompting many to state that medical institutions are data rich but information poor.

The potential of data mining in this area has been recognized by many. Several efforts are underway to develop systems and analytics able for the modeling of patient states and the early detection of complications. In general, early detection of complications can lead to earlier interventions or prophylactic strategies to improve patient outcomes. Early detection rests on the ability to extract subtle yet clinically meaningful correlations that are often buried within several multi-modal data streams and static patient information, spanning long periods of time.

Systems for data mining in intensive care:

Modern patient monitors have evolved into complex system that not only measure physiological signals but also produce alerts when the physiological state of the patient appears to be out of range. State of the art patient monitors allow physicians to program thresholds defining normality ranges for physiological systems. For example, one can program a patient monitor to produce an audible alert if the oxygen saturation level of the blood is below 85 percent. The values of these thresholds are typically obtained from general guidelines or from data mining processes. Such simple alerting schemes are well known to produce very large numbers of false alarms. In [23], it is reported that more than 92 percent of alarms generated in an ICU are of no consequence. Furthermore, there are many complex physiological patterns of interest to physicians that cannot be represented by a set of thresholds on sensor data streams. Several research initiatives are addressing this problem with the design of platforms facilitating analysis beyond the simple thresholding capabilities of existing patient monitoring systems.

One example is BioStream [24], a system that performs real-time processing and analysis of physiological streams on a general purpose streaming infrastructure. The authors use ECG data along with temperature, oxygen saturation, blood pressure and glucose levels as inputs into patient-specific analytic applications. The system supports a different processing graph (for analysis) per patient, where the graph can be composed of system supplied operators (functions) and user implemented operators. The authors also state that BioStreams can be used to discover new patterns and hypotheses from the data and test them, however there is limited discussion of the underlying analytics and use cases.

In [25] the authors describe an architecture for a system whose goals are data mining, fusion, and management of data streams for intensive care patients. The proposed system has online components for capture of physiological data streams and program execution along with off-line components for data mining.

The SIMON (Signal Interpretation and MONitoring) platform [26] developed at Vanderbilt is a data acquisition system that continuously collects and processes bedside patient monitoring data. SIMON collects typical ICU monitoring vital signs including heart rate, blood pressures, oxygen saturations, intracranial and cerebral perfusion pressures, and EKG-ECG waveforms. This data collection is intended to support clinical research by enabling further analysis and mining. The system is also

capable of producing alarms and has reporting capabilities though a web interface and through event notification mechanisms.

The Online Healthcare Analytics infrastructure, which is also known as Artemis [27], is a programmable framework for real-time analysis of intensive care sensor data leveraging the IBM InfoSphere Streams (Streams) real-time high-performance stream analysis engine. OHA interfaces Streams with an open set of data collection systems (e.g., Excel Medical Electronics BedMasterEX system, the CapsuleTech data collection system), and leverages different data mining technologies and machine learning algorithms for the generation of models for prediction of the onset of complications in intensive care. OHA leverages Streams interface with well known analytic softwares such as SPSS, SAS and R to provide data mining capabilities. Models learned with these data mining systems can be scored in real-time, thus giving the analyst-physician the ability to test clinical hypotheses prospectively. This analytical loop is abstracted in Figure 14.2. It constitutes a general architecture for sensor data mining applications leveraging both at rest analytics for modeling and in motion analytics for the scoring of models in real-time. The OHA system has been in use in live environments for the monitoring of neonates [27]. Its exploration capabilities are also used for the mining of sensor data for the early detection of complications in neurological ICUs [29].

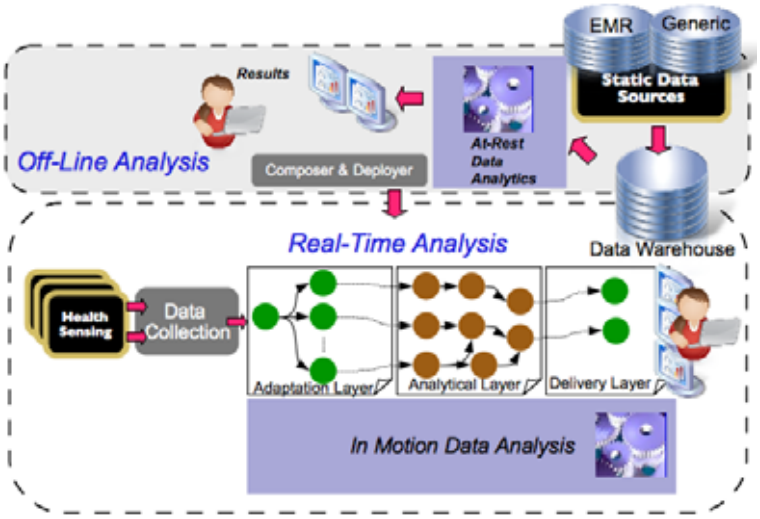


Figure 14.2. A generic architecture for sensor data mining systems

OHA has been extended with patient similarity concepts to help physicians take decisions while leveraging past experiences gathered from similar patients that have been monitored in the past [30]. In [31], the MITRA system, introduced as an extension of OHA, allows physicians to query for similar patients and use records from these similar patients to make predictions on the health evolution of a patient of interest. An in-silico study using physiological sensor data streams from 1500 ICU patients obtained from physionet [32] shows how MITRA may be used to forecast the trajectory of blood pressure streams and help predicting acute hypotensive episodes in ICUs. In [33], similar approaches to time-series forecasting with applications to intensive care are also reported. Patient similarity techniques are described in this thesis as a way to extract robust features for forecasting purposes. Sequential learning techniques with Linear Dynamical Systems and Hidden Markov Models are proposed for the modeling stages.

State-of-the-art Analytics for Intensive Care Sensor Data Mining: State of the art analytics and mining approaches for in hospital sensor data monitoring tend to generate innovations on data pre-processing and transformation. Modeling is typically done with well known families of machine learning techniques such as classification, clustering and dynamic system modeling with sequential learning. These analytical techniques often attempt to derive features from physiological time series to model the *inflammatory response* of the body, as it is known to be highly correlated with early sign of complications in general. The inflammatory response is a reaction from the body to different harmful stimuli such as pathogens, various irritants or even damaged cells. Hence, accurate modeling of it enables a wide range of early detection applications in intensive care. In particular, devastating complications such as sepsis are known to produce an inflammatory response well before the appearance of clinical symptoms [51].

The inflammatory response is controlled by the autonomic nervous system, consisting of the sympathetic and parasympathetic nervous systems [40]. These systems regulate several involuntary actions such heart beats, respiration, salivation, transpiration etc. Inflammation results in poor regulation of these systems, and is often correlated with the Systemic Inflammatory Response Syndrome (SIRS) [41], [42]. The poor regulation manifests itself in loss of signal variability associated with physiological sensor streams. As a result, several researchers have attempted to model the inflammatory response using various measures

estimating the signal variability of heart rate observations[2]. Monitoring reductions in Heart Rate Variability (HRV) has been a successful strategy for the early detection of disorders of the central and peripheral nervous system that induce a pro-inflammatory response [43].

Figure 14.3. Sensing in intensive care environments

Existing efforts to model the inflammatory response are focused primarily on the one dimensional HRV analysis, due to a large body of work on ECG waveform processing. The Society for Complexity in Acute Illness (SCAI) [46] has devoted many efforts to model complexity and variability in the human body from ECG signals, as a way to model ICU patients and derive models predicting complications in ICUs. Variability metrics [47] typically used include spectral analysis techniques [52], approximations to uncomputable notions of randomness with the approximate and sample entropy [48],[44], and fractal analysis techniques like the Detrended Fluctuation Analysis [49]. Surprisingly, classical information theoretic approaches to measure complexity with well understood concepts of compressibility and predictability [50] have received a modest amount of attention in acute care.

[2]Reductions in the variability of other vital signs such as respiration may also be correlated with the inflammatory response.

The success of variability analysis has been reported by many researchers. In [51], the authors perform a spectral analysis of heart rate measurements to show a relationship between heart rate variability and sepsis. In [47], the potential of this approach is highlighted with a description of multiple clinical applications that use such complexity analysis. In [52] the authors derive several empirical links between heart rate variability and mortality in intensive care units. In [53], the prognostic potential of heart rate variability measures in intensive care is proposed. The authors in [54] have shown that reductions of heart rate variability are correlated with outcomes in pediatric intensive care. At the University of Virginia, Lake [45] et. al. have used the sample entropy on heart rate measurements to predict the onset of sepsis in neonates. In [55] the predictive capability of heart rate variability on the prognosis of a large population of trauma patients is described.

Heart rate variability has also been use to determine when to extubate or remove patients from mechanical ventilation in intensive care [56]. A clinical trial is currently underway in Canada testing whether maintaining stable heart rate and respiratory rate variability throughout the spontaneous breathing trials, administered to patients before extubation, may predict subsequent successful extubation [56].

Besides heart rate variability analysis, there are many other applications of sensor data mining intensive care. Analysis of the dynamics of the ECG signal has enabled researchers to build systems for arrhythmia detection using standard machine learning and classification techniques. The work presented in [57] is illustrative of these systems.

Respiratory complications have also received a significant amount of attention in the intensive care community. In [58] the authors describe the use of sensor data from brain activity measured with electroencephalograms (EEG), eye movements measured with electrooculogram (EOG), muscle activity measured with electromyogram (EMG) and heart rhythm measured with ECGs during sleep, to detect obstructive sleep apnea episodes, that are known to be correlated with poor patient outcomes.

EEG signals have also been used beyond sleep apnea studies. In [59], EEG spectral analysis is performed to detect epileptic seizures with machine learning techniques, while in [60], continuous EEG spectral analysis for brain ischemia prediction is illustrated.

General predictive models for patient instability in intensive care have also been proposed in the literature. A notable example is the work in [61], where the authors extract several time series trending features from heart rate and blood pressure measurements collected every minute and build predictive models using a multi-variable logistic regression mod-

eling algorithm. This simple approach proves the ability to generate predictive alerts for hemodynamically unstable patients with high accuracy from trends computed on physiological signals.

In [62], a belief Bayesian belief network is developed to model ICU data and help care givers interpret the measurements collected by patient monitors. The belief-network model represents knowledge of pathophysiologic or disease states in a causal probabilistic framework. The model is able to derive a quantitative description of the physiological states of the patients as they progress through a disease by combining the information from both qualitative and quantitative or numerical inputs.

Another relevant body of work on sensor mining in intensive care environments has focused on the identification and removal of undesirable artifacts from sensor data streams. This includes mitigating the impact of missing and noisy events, as well as clinical interventions (e.g. drawing blood, medications) that complicate the data mining process (Section 2). In [63], a factorial switching Kalman Filtering approach is proposed to correct for artifacts in neonatal intensive care environments. In [64] the authors develop clever techniques leveraging dynamic Bayesian networks to analyze time series sensor data in the presence of such artifacts.

3.2 Sensor Data Mining in Operating Rooms

Data mining applications that relate to operating rooms tend to focus on the analysis of Electronic Medical Record data where most sensor data inputs are filtered and summarized. For example, in [22], EMR data is used to improve the efficiency of operating rooms, in terms of scheduling (start times, turnover times) and utilization. In [21], knowledge management and data mining techniques are used to improve orthopedic operating room processes, yielding more effective decision making.

A few researchers have reported applications directly mining physiological sensor data produced by operating room monitoring systems. Exceptions are presented in [65] where the authors correlate EEG signals with cerebral blood flow measurements for patients undergoing carotid endarterectomy. This finding is quite valuable as it proves that EEG signals can be used to monitor complex mechanisms including cerebral blood flow for this patient population. In [66], machine learning techniques are proposed for the closed loop control of anesthesia procedures. In [67],the authors present a prototype of a context-aware system able to analyze patient data streams collected in an operating room during surgical procedures, to detect medically significant events and persist them in specific EMR systems.

3.3 General Mining of Clinical Sensor Data

In general clinical settings, data mining is often confine to the mining of Electronic Health Records (EHR) that do contain sensor data (e.g., patient vital signs). With the capture of patient medical histories into EHRs and the strong push worldwide to introduce EHRs in healthcare systems, systems capable of mining these data are receiving more and more attention. EHRs are data rich. They include structured and unstructured comprising of all the key administrative clinical data relevant to patients, demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data, diverse test results and radiology reports [16]. Unfortunately, there are no widely accepted standards for the representation of all these data points stored in EHR systems. Several code systems (e.g., ICD-9, ICD-10, CPT-4, SNOWMED-CT [34]) and interoperability standards (e.g., HL7, HIE) are in use by many systems but there are no overarching standards that EHR vendors are adhering to. Despite this lack of global standardization that is hindering the realization of very large scale data mining, many researchers are spending considerable efforts to analyze these data sets to improve healthcare in general.

In [16], EHR data are mined to derive relationships between diabetic patients usage of healthcare resources (e.g., medical facilities, physicians) and the severity of their diseases. In [17], Reconstructability Analysis (RA) is applied to EHR data to find risk factors for various complications of diabetes including myocardial infarction and microalbuminuria. RA is an information-theoretic technique used for mining of data sets of large dimensionality. In this setting, RA is used to induce relationships and correlations between EHR variables by identifying strongly related subsets of variables and to representing this knowledge in simplified models while eliminating the connections between all other weakly correlated subsets of variables.

In [18], data quality issues are reported while attempting to analyze EHR data for a survival analysis study on records of pancreatic cancer patients. Incomplete pathology reports for most of these patients forced the authors to exclude them from their study. The authors conclude this paper by suggesting complementing EHR data with more generic patient related data to produce more complete patient representations where such data mining studies can be performed.

Batal et. al. present in [19] an approach to find temporal patterns in EHR data. At the core of their technique is the representation of longitudinal patient records with temporal abstractions. These abstractions are essentially summaries of intervals of time series data. For example,

patient body mass indices may be abstracted by increasing-decreasing-steady trend qualifiers. The authors also propose techniques for mining such EHR temporal abstraction using standard data mining schemes (e.g., apriori algorithm).

Neuvirth and his colleagues [35] proposed an interesting application of data mining techniques on EHR data for the management of chronic diseases. This application is able to predict patient future health states and identify high risk patients for specific diseases where risk is a function of the likelihood of needing emergency care and the likelihood of receiving sub-optimal treatments. They further explore the links between physicians treating these patient populations and outcomes to design a system that optimizes the matching between individual patients and physicians for better outcomes. Their analysis makes heavy use of standard machine learning techniques (e.g., logistic regression, K-Nearest Neighbor classification) and survival analysis (Cox modeling) and has generated interesting results for the management of diabetic patients.

The concept of patient similarity described above in Section 3.1.1 has also been on EHR data with the AALIM system [36] which uses content based search techniques on different modality data to extract disease specific patient information and find groups of similar patients. AALIM uses data from similar patients to help physicians make prognosis for a given patient and design care management strategies. Sensor data inputs into AALIM includes ECGs, videos, echocardiograms, MRIs and text notes.

With the emergence of question answering systems like IBM Watson [37], the potential to design systems able to ingest very large amounts of structured and unstructured clinical data to support clinical diagnosis and prognosis is emerging. The ability of Watson to analyze the meaning and context of human language, and quickly process vast amounts of information to answer questions has wide applicability in healthcare. One can imagine applications where a properly trained Watson system can assist decision makers, such as physicians and nurses, in identifying the most likely diagnosis and treatment options for their patients. IBM and Wellpoint have partnered to develop such a system with applications to patient diagnosis [38]. A similarly partnership with Memorial Sloan Kettering is in place for the diagnosis and management of cancer [39]

4. Non-Clinical Healthcare Applications

The world is experiencing a rapid increase in its aging population, and a corresponding increase in the prevalence of chronic diseases and health care expenditure. For instance, the total Medicare expenditure in

the United States has risen from $239.5 billion in 2000 to $524 billion in 2010, and this is likely to continue significant growth for the foreseeable future. *Aging in place* has been proposed as one method to reduce cost and maintain quality of life for the aging population. The concept is to support older adults in the environment of their choice as opposed to placing them in traditional clinical settings or or nursing home environments. Healthcare is being looked at as a continuum expanding outside of traditional clinical settings with goals to make it more proactive to reduce stress on medical institutions. Providing healthcare support outside of clinical environments with smart monitoring devices and e-health technology has been the focus of much research recently, specially in the ubiquitous computing research community.

Ubiquitous healthcare [71] is an emerging field of research that uses a large number of environmental and body sensors and actuators combined with sensor mining and analytic technologies to monitor and improve health of people in these types of settings. Ubiquitous healthcare approaches often employ several distributed and wireless sensors to gather information on bodily conditions such as temperature, heart rate, blood pressure, blood and urine chemical levels, breathing rate and volume, activity levels, and several other physiological characteristics that allow diagnosis of health problems. These sensors are often worn on or implanted in the body, or installed in the environment. Additionally, these sensors also include actuators that can trigger actions such as the release of small quantities of pharmaceutical drugs into the bloodstream, or the electrical stimulation of brain areas (e.g. those implicated in conditions such as Alzheimers disease and Parkinson disease or those associated with depression). Finally, there are also non-intrusive, but wearable sensors that capture the motion of the body during its execution of different activities. Research in the wearable computing community has shown that characteristic movement patterns for activities such as running, walking or lying can effectively be inferred from body-worn accelerometers.

Ubiquitous healthcare has also relied heavily on the construction of *smart environments* where the environment itself is instrumented to capture the user behavior and their interaction with the external world. This includes several Radio Frequency Identification (RFID) tags and readers because of their durability, small size, and low costs. There is significant use of infrared sensors as well as video cameras and other sensors for motion detection, image processing, and control of in-home devices. Some environments also employ ultrasonic location tracking sensors, pressure sensors (deployed in various surfaces such as floors

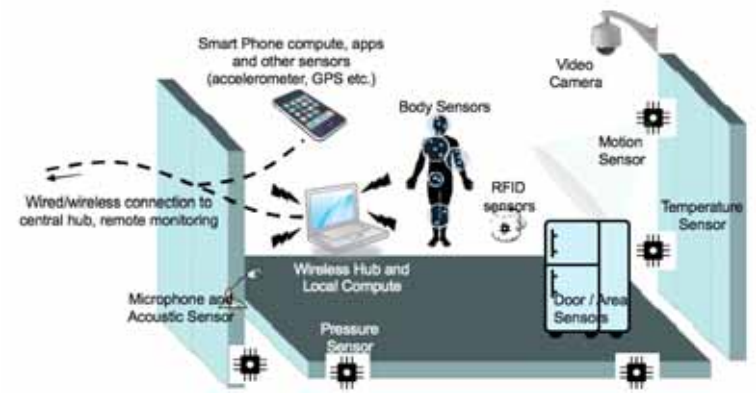


Figure 14.4. Sensing in home environments

etc.), and smart displays for information dissemination. These sensors are embedded in different parts of the home and workplace environment including on doors, beds, mirrors, bathrooms, mailboxes, in appliances such as microwaves and allow determining a comprehensive picture of user activities.

There are several tradeoffs that need to be considered when deciding how many *smart environment* sensors are needed and where they should be placed in order to provide enough information for the analysis to accurately recognize activities. While a greater density of sensors provides more accurate information on the person position and their interactions with the environment, this comes with increased energy consumption, cost constraints, and intrusiveness. In addition, increasing sensors lead to increasing complexity, thus requiring a greater amount of data, large-scale algorithms, and systems to accurately learn activity models.

Reality mining [72] is also an emerging field complementing ubiquitous healthcare and leveraging data mining technologies. Reality mining processes all digital information available in the daily environments in which we evolve these days. Many of the daily activities we perform, such as checking our email, making phone calls, making a purchase, commuting etc., leave digital traces and can be mined to capture records of our daily experiences. These human physical and social activity traces are captured by the multitude of sensors in mobile phones, cars, security cameras, RFID (smart card) readers, road-transport sensors etc. Reality mining [72], is an emerging field of research that uses statistical analysis and machine learning methods on these digital traces to develop comprehensive pictures of our lives, both individually and collectively. Computational models based on this data, combined with any physio-

logical information collected from body sensors and smart environments, can dramatically transform both individual as well as community health.

The different healthcare applications in non-clinical settings that we address in this chapter may be broadly categorized into:

- **Chronic Disease and Wellness Management Applications** that facilitate preventive care and chronic disease management and treatment, along with user programs to motivate happy and healthy behavior.
- **Activity Monitoring Applications** that capture activities of daily living especially for elderly users, in remote healthcare settings
- **Reality Mining applied to Healthcare** that applies machine learning techniques to data typically sensed with mobile phones to study complex social systems, including the study of the distribution and patterns of health-events, health-characteristics and their causes or influences in specific populations.

4.1 Chronic Disease and Wellness Management

Several researchers have reported on remote patient monitoring systems with sensor mining capabilities for chronic disease and wellness management. In [28] the authors report on an interesting prototype streaming system called T2, designed to monitor mobile patient ECGs and accelerometers data streams, remotely. The application reports periods of elevated heart rate to the clinician. The accelerometer is used to detect periods of physical activity during which ECG data is filtered to account for different activity levels.

Holter monitors constitute another class of sensors that are worn by a patient continuously for several days to provide a complete ECG reading over that time. The analysis of the recorded data is done offline to detect cardiac conditions of interest. The use of Holter Monitors is expanding as researchers seek ways to detect conditions and treat patients who have multiple diseases. In [68], researchers record both glucose and cardiac readings in diabetes patient with cardiac conditions to detect correlations between high glucose readings and ECG patterns.

A remote monitoring platform called Personal Care Connect (PCC) [69] has been extended with advanced distributed analytical capabilities. The resulting Harmoni platform allows for the distribution of analysis from back-end servers to remote devices located near the patient. In addition, the Harmoni platform allows for the distribution and instantiation of monitoring rules, triggered by changes in the context of the

user being monitored. For example, while monitoring the heart rate of the patient, thresholds for what constitute a normal heart rate are adjusted by inferring the activity of the patient (e.g., sitting down vs. walking). The battery power of the sensors are extended with analytical rules requiring sensors to report measurement in a more granular way in emergencies or during abnormal physiological episodes [70].

Several Wireless Body Sensor Networks (BSNs) [73] [74] have been used in several pilot applications for monitoring elderly patients with chronic conditions in out-patient settings. Using medical sensors such as ECG, several cardiovascular related illnesses can be detected as early as possible by simply monitoring heart-beat rhythm (arrhythmias). Multiple heterogeneous sensor architecture can help expending the boundaries of BSNs application ranges. The DexterNet BSNs [75] use motion sensors (motes), GPS, and airborne particulate matter (PM) sensors to monitor as well as prevent asthma. Motion sensors (Accelerometer) combined with with Electromyogram (EMG) sensors that capture human motion balancing and muscular actives have been used to build postural stability and subject-independent classification models.

Recent work on using motion sensors with other body sensors is included in the myHealthAssistant [76] project. The focus of this project is on preventive health care and development of a system that helps reducing physical inactivity. The system captures individual activity through the day, and motivates users by calculating a new workout plan based on completed workouts[3]. In the base setup for daily activity monitoring, a set of a single customized accelerometer, a smartphone, and a heart rate sensor are used to identify five different activities, monitor the heart rate and calculate the calorie expenditure. The system also allows the user to wear two additional accelerometers (strapped around the torso, and attached to the right weight lifting glove) while exercising in order to get a more accurate identification of 16 activities, and calorific expenditure. The analysis is performed at a local computer using a Gaussian model-based classifier.

Many other body sensor applications have been designed to monitory physical activity[4] as it is critical to maintain physical and psychological health, and reduce the risk of premature mortality, coronary heart

[3]Several studies [77] have shown that internet and phone based user motivation systems can significantly increase the level of physical activity.

[4]Commercial systems to encourage physical activity are used only while performing the target activity and are not trying to disambiguate that activity. Such technologies include Dance Dance Revolution, the Nintendo Wii Fit, the Nike+ system, Garmins Forerunner, Bones in Motions Active Mobile and Active Online, bike computers, heart rate monitors, MPTrain [17], Jogging over a distance [15], and mixed- and virtualreality sports games [13,14].

disease, type II diabetes, colon cancer, and osteoporosis, and symptoms associated with mental health conditions such as depression and anxiety. Researchers [78] have developed the UbiFit Garden, which uses on-body sensing, activity inference, and a novel personal, mobile display to encourage physical activity. The UbiFit Garden system consists of three components: a fitness device, an interactive application, and a glanceable display. The fitness device automatically infers and communicates information about several types of physical activities to the glanceable display and interactive application. The interactive application includes detailed information about the individuals physical activities. The glanceable display, that resides on the background screen of a mobile phone uses a non-literal, aesthetic representation of physical activities and goal attainment to motivate behavior. The UbiFit application includes the continuous monitoring of different fitness parameters and building statistical models of these to compute and project trends, and provide better information to users. Several other such fitness and physical activity monitoring applications are presented in [73].

The authors in [79] have shown how body movements and eye movements can be used to provide contextual information to an adaptive hearing instrument to distinguish different hearing needs in various acoustic environments. The authors record body movements, eye movements (using electrooculography), and hearing instrument sound in different simulated acoustic environments. They then use an SVM based classifier and person-independent training to show that these different sensor readings can be used to accurately (in some cases up to 92%) determine the acoustic environment characteristics, and modify the settings of the hearing instrument appropriately.

Correlating different body sensors to monitor dietary activities has been demonstrated in [84]. The authors capture dietary parameters such as the rate of intake (in grams per seconds.) the number of chews for a food piece etc. that capture palatability, satiety and speed of eating. In particular, three core aspects of dietary activity were investigated using sensors: characteristic arm and trunk movement capture using inertial sensors, chewing of foods and food breakdown sounds using an ear microphone, and swallowing activity using a sensor-collar containing surface Electromyography (EMG) electrodes and a stethoscope microphone. The authors then build a recognition algorithm using time and frequency-domain features that addresses multiple challenges of continuous activity recognition, including the dynamic adaptability for variable-length activities and flexible deployment by supporting one to many independent classes. The approach uses a sensitive activity event search followed by a selective refinement of the detection using

different information fusion schemes. The authors use selective fusion of detection results exploiting independent sources of error to filter out false positives and obtain an event classification in the same step, and achieve highly accurate activity recognition.

Recent work [85] has also focused on the use of body sensors for patient authentication. Credential based authentication methods (e.g., passwords, certificates) are not well-suited for remote healthcare as these may be compromised. One-time authentication using credentials or trait-based biometrics (e.g., face, fingerprints, iris) do not cover the entire monitoring period and may lead to unauthorized post-authentication use in some situations. Recent studies have shown that the human electrocardiogram (ECG) exhibits unique patterns that can be used to discriminate individuals. However, perturbations of the ECG signal due to physical activity in real-world situations can lead to authentication failures. The authors in [85] build an activity-aware biometric authentication system that combines ECG information with accelerometer data to handle the variability that arises from physical activity. The authors use the SHIMMER [86] sensing platform (with an integrated 3-axis accelerometer) developed by Intel Digital Health Advanced Technology Group to combine the motion and activity data with the ECG signal using a direct cable to a commercially available Polar WearLink Plus ECG chest strap. The sensor data is transmitted via Bluetooth device to a commputer running the BioMOBIOUS software for analysis. The mining uses different types of feature cleaning and preprocessing (beat-based linear interpolation)combined with K-Nearest Neighbor (KNN) and Bayesian network(BN) classification to obtain accurate user authentication under different activity levels.

The MIThril [87] project is focused on developing a next-generation wearable sensor research platform. The project includes the development and prototyping of new techniques of human-computer interaction for body-worn applications, through the application of human factors, machine learning, hardware engineering, and software engineering. The MIThril project also involves research into constructing a new computing environment and developing prototype applications for health, communications, and just-in-time information delivery. The MIThril LiveNet [88] is a flexible distributed mobile platform that can be deployed for a variety of proactive healthcare applications. The LiveNet system allows people to receive real-time feedback from their continuously monitored and analyzed health state, as well as communicate health information with care-givers and other members of the social network of an individual for support and interaction. Key components of this system include a PDA-centric mobile wearable platform, the Enchantment soft-

ware network and resource discovery API, and the MIThril real-time machine learning inference infrastructure. The LiveNet system is currently in use for multiple studies: capturing the effects of medication on the dyskinesia state of Parkinsons patients [89], a pilot epilepsy classifier study with the University of Rochester Center for Future Health, a depression medication study with the MGH Department of Neuroscience, and a hypothermia study with the Advanced Research in Environmental Medicine (ARIEM) at the Natick Army Labs [90].

The MyHeart [91] project funded by the IST program of the European Commission is a concerted effort aimed at developing intelligent systems for the prevention and monitoring of cardiovascular diseases using smart electronic and textile systems based wearable sensors, and appropriate services that empower the users to take control of their own health status. The MyHeart project integrates functional clothes with on-body sensors (textile and nontextile) and electronics to acquire, process and evaluate physiological data. It also includes a wireless personal area network to transmit results to a mobile phone or PDA and from there to a server farm, to request professional medical services. Recently, there have also been several developments that combine on-body sensors with implantable sensors. The Healthy Aims [92] project of the European Commission focuses on developing a range of medical implants (Cochlear implant, retina implant and glaucoma sensor, implantable pressure sensor to monitor intracranial pressure, Sphincter sensor, and Inertial Measurement Unit) to assist aging people with disabilities.

The Wealthy [93] consortium was also established by the European Commission to fulfil the need to continuously monitor patient vital signs through novel woven sensing interfaces that could be worn without any discomfort for the user. The focus of the project is on development of smart material in fiber and yarn form endowed with a wide range of electrophysical properties (conducting, piezoresistive, etc) for use as basic elements. The Alert Portable Telemedical Monitor (AMON), is another project whose aim is to develop a wrist worn device encapsulating many sensors. Currently, blood pressure, pulse oximetry, ECG, accelerometer, and skin temperature are available. The device communicates directly to a telemedicine center via a GSM network, allowing direct contact with the patient if necessary. AMON enables patients which are not confined to a hospital to monitor continuously and analyze their vital signs.

The Motion Analysis Lab [94] is focused on researching rehabilitative tools in the treatment of mobility-limiting conditions in people with cerebral palsy, stroke, traumatic brain injury, spinal cord injury, Parkinsons Disease, and other neuromuscular disorders. In pursuit of this goal, the MAL focuses on the rehabilitative possibilities of robotics and wearable

sensor technology. The lab adopts these technologies for the purposes of retraining gait in children with cerebral palsy and is leading research into development better prosthetics for amputees, interactive technology for stroke survivors, and traumatic brain injuries and people with burn-related contractures.

There is emerging interest in building Body Area Sensor Networks - large-scale BSNs across a public healthcare system such as a hospital. The miTag system [95] is a pilot public healthcare BSN deployed in the Baltimore Washington Metropolitan region. This system includes a wireless multi-sensor platform that collects information from GPS receivers, pulse oximeters, blood pressure cuffs, temperature sensors, and ECG sensors. The system supports two-way communication between patients and healthcare providers, to allow for feedback based on the monitored health and context information. Body Area Sensor Networks are also being developed to support disaster management in emergency response systems.

The maturity of sensor networks has allowed the development of smart environments for wellness and chronic disease management. For example, some researchers have used smart environments with combinations of wearable devices (RFID bracelets) and RFID tagged objects to detect indications of cognitive impairments such as dementia and traumatic brain injury (TBI) by monitoring individuals performing a well defined routine task - making coffee [103]. The researchers define and compute a set of four domain specific features from the sensor data, that are increasingly representative of the task, and correlate with severity of cognitive impairment. These features include the Trial Duration, Action Gaps, Object Misuse, and Edit Distance. Trial Duration captures the total time taken for the activity while Action Gaps represent periods during which subjects were not interacting with any objects on the assumption that during those periods they are considering what step to take next. Object Misuse captures the number of times a subject interacts with each object used in the task - with failure to interact with a required object, or an excessive number of interactions indicates problems. Finally, the researchers manually define a representative plan[5] for the task, that represents a partial order (to allow alternate reasonable task executions) over object interaction. The Edit Distance, as used in natural language processing then captures deviations from this plan. Finally, these features are analyzed using Principal Component Analysis (PCA) to examine correlations between computed features and larger trends in the

[5]Other research on activity recognition has addressed the question of automatically constructing plans for everyday activities by mining the web for descriptions of these activities.

assessment data. They show that the first principal component includes a diverse set of measures of general intelligence, and appears to be a good proxy for general neuropsychological integrity, including measures of intellectual functioning, verbal and nonverbal reasoning, memory, and complex attention.

Researchers are developing several other techniques for the automatic detection of cognitive impairments, including automatically observing users play modified versions of different games. For instance, a modified version of the game FreeCell [104] is used in many studies. One study focuses on mouse movement during the game while others focus on the subject performance over time, comparing it to the performance of an automated solver. Using the results, it was possible to differentiate the three mildly cognitively impaired subjects from the six others. Work with several other computer games, specially created to perform assessments of cognitive impairments is underway with some promising early results. Researchers have also studied automatically monitoring mobility because slowed mobility may be a predictor of future cognitive decline. The time to answer a phone call was used to measure mobility, as were passive infrared detectors and several models to infer the mobility of subjects more directly as they move about a residence. More details on these may be obtained from [103].

Mining data from smart environments has also been used for sleep research [105] on a long-term basis, in a comfortable setting[6]. Inertial, ambient light, and time data are tracked from a wrist-worn sensor, and additional night vision footage is used for later expert inspection. The authors use two different classification techniques to monitor and classify the night sleep. Classifier 1 use threshold-based segmentation on a Gaussian model-based classifier that calculates the variance and mean parameters for the light intensity and motion data from the training data, and uses a likelihood per minute of the awake state from the time-use database. Classifier 2 uses HMM-based segmentation to capture changes in sleep habits and state, and differentiate awake state from sleep state. The authors have shown that these techniques can be used for accurate sleep studies while minimizing the intrusiveness of the sensing environment for patients suffering from sleep disorders and psychiatric illnesses.

[6]The golden standard for observing sleep-wake patterns is polysomnography (PSG) that captures relevant sleep information with typically 20, mostly wired sensors attached to the face, torso and limbs of the patient, making it costly, uncomfortable and less feasible over longer periods.

There has been a fair amount of work on using smart environments combined with body sensors for personal cardiac monitoring. This includes projects like Mobihealth [107] and PhMon [106]. Many of these solutions collect the physiological signals, but ECG analysis is performed remotely after transmission over a GPRS network. Recent work in multiple projects has enabled the processing of ECG data on a local device. MOLEC [108] analyses the ECG locally on a PDA and generates alarms to the hospital in case of high risk arrhythmias. The authors in [109] develop an application whereby a heart patient is monitored using various types of sensors (ECG, accelerometer, Oxygen), and analyzed locally on a smart phone. The solution can be personalized by capturing location context, and includes rehabilitation applications for individual patients.

In addition to smart environment and body sensors, there are also efforts at building platforms for wellness management. One such platform is Greenolive [110], an open scalable platform providing services that are essential to wellness management. Greenolive includes open APIs that allow new value-added applications to be developed rapidly. The Greenolive platform consists of four components: Data Transformation and Routing Services, Wellness Monitoring Services, Wellness Analytic Services and Wellness Record and Knowledge Repository. With these components, using a cloud based compute infrastructure, developers can create different portals targeted towards both care assistants as well as portals that connect with the sensors and provide end users wellness services. More details on the platform and the included mining and analytic capabilities are presented in [110].

4.2 Activity Monitoring

Several smart environments [96] have been built, deployed and tested for pervasive healthcare applications focusing in activity monitoring. These smart homes or offices include combinations of environmental sensors, embedded in the home or the external environment, and body sensors for improved monitoring of people with different conditions and healthcare requirements.

One of the key roles of smart environments is to enable researchers to monitor activities of daily living (ADL). In order to function independently at home, individuals need to be able to complete several activities of daily living such as eating, dressing, bathing, cooking, drinking, taking medicine etc. Automating the recognition of these activities is an important step toward monitoring the functional health of a smart home resident. In addition to the ADL, researchers are also very interested in the interactions of users with the physical and social environment.

This includes another set of activities such as using a telephone, shopping, housekeeping, doing laundry, transportation, handling finances etc. These are collectively labeled Instrumental Activities of Daily Living (IADL) and also indicate different aspects of the functional health. In the absence of smart environments, the assessment of ADLs and IADLs has mostly been done manually through interviews and questionnaires. This is often a very time consuming and error prone process, and hence there is a strong need to automate the monitoring and recognition of these ADL-IADLs continuously via smart environments. We describe several of these smart environments and their applications in the following paragraphs.

Some smart homes with healthcare technology for older adults have been developed as part of a laboratory setting. The Smart Medical Home at University of Rochesters Center for Future Health [97] is one such example. The five-room house has infrared sensors, computers, biosensors, and video cameras. A large part of the research involves interactions of the research subjects (patients) with a medication advisor who provides advice on medication management and dietary adherence, memory assistance, and assistance with Smart Bandage. Smart Bandage is a program designed to decrease the burdens of chronic wound care at home. Future applications of this laboratory environment include gait monitoring, and observation of behavior and sleep. The Smart Medical Home is designed for adults of all ages, but it is not meant for actual habitation.

As described in [96], the Gator Tech Smart House at the University of Florida-Gainesville Mobile and Pervasive Computing Laboratory [99] is a laboratory-house created to assist older adults in maximizing independence. The house is equipped with (a) smart cameras for motion detection, image processing, and control of other in-home devices, (b) smart blinds that automatically close to block sunlight when the air conditioner is on, (c) ultrasonic location tracking transceivers that are installed on the ceiling corners of each room to detect movement, location, and orientation of the resident, (d) smart floor that uses pressure sensors embedded into each tile to detect falls and reports to emergency services, and (f) smart displays for entertainment media and information residents can follow from room to room. The house also includes a smart mailbox that senses and notifies the arrival of mail, a smart front door that identifies residents, using a radio-frequency identification tag among others, a smart bed that monitors sleeping patterns, a smart mirror that displays important messages or reminders such as when to take medication, and a smart bathroom that includes a toilet paper dispenser, a flush detector, and a water temperature regulating

shower. The Gator Tech Smart House is adding healthcare technologies to assist diabetes management.

A set of smart home environments called CASAS has been setup in Washington State University. The CASAS home has five different testbed environments. The first, referred to as Kyoto [100], is a two-bedroom apartment that is equipped with motion sensors (positioned on the ceiling 1 m apart throughout the space), sensors to provide ambient temperature readings, and custom-built analog sensors to provide readings for hot water, cold water, and stove burner use. Voice over IP captures phone usage, contact switch Q4 sensors monitor the open-closed status of doors and cabinets, and pressure sensors monitor usage of key items such as the medicine container, cooking tools, and telephone. The second testbed, referred to as Cairo is a two-bedroom, two-story home. There are three additional environments configured as single-resident apartments (Bosch1, Bosch2, and Bosch3) that are part of a single assisted care facility. All of these environments contain motion sensors throughout the space as well as door contact sensors in key areas. Sensor data for each of the environments are captured using a sensor network and stored in a database. The data is analyzed for automatic ADL recognition, monitoring of diabetic patient diet, and exercise adherence. These environments also allow the presence of pets along with humans to simulate realistic settings. Researchers employ Hidden Markov Models (HMMs) to recognize possibly interleaved activities from a stream of sensor events, with the hidden states representing activities. There is also strong emphasis on questions pertaining to the selection, placement, and focus of sensors in a smart environment. In several studies conducted by researchers [100], they have employed mutual information (MI) based measures to rank sensors, and quantify the mutual dependence between the sensor reading and the activity of interest. They then use a filter-based sensor selection strategy to systematically evaluate the effect of removing sensors with low MI values on activity recognition performance. They also use hierarchical clustering to identify sensors with overlaps in the field of view in order to remove unnecessary sensors, and determine appropriate placements for the deployed sensors using a decision tree learner. They have shown that reductions on average of 20 percent of the sensors are possible for different types of activities and different configurations of the smart home.

Other examples of laboratory smart environments include a two-story single-family house called Aware Home developed by the Georgia Institute of Technology. This is a living laboratory house designed primarily to assist adults with cognitive impairment [98]. For instance, the home includes a capture system on the kitchen countertop with a wall display

that shows visual snapshots arranged as a series of panels to enable review of activities for users. A similar system can be used to support safe and complete medication adherence. This technology has also been used for diabetes management using a mobile phone to which a glucose meter can be connected via Bluetooth.

Besides these laboratory settings, there are also several smart homes that have been implemented in actual community settings, apartment complexes, and retirement housing units. These include a smart home in Vinson Hall Retirement Community in Missouri that is dedicated to serving former U.S. military officers and their families. Eskaton, Ltd. has created the National Demonstration Home in California with a range of technologies. The University of Missouri-Columbia has integrated sensor networks into privately owned apartments called TigerPlace II. A community wide comprehensive smart home deployment is under development in McKeesport, Pennsylvania. The University at Buffalo, State University of New York, has utilized X10 devices to retrofit 50 homes for older adults with chronic conditions living alone in their own home. More details on these and other such smart home projects can be obtained from [96].

Researchers have recently investigated the use of domestic robots as a promising technology for persuasive telehealth [101]. Domestic robots have several unique features as compared against other devices in smart environments. One reason some technologies are difficult to use in persuasive telehealth systems is because they require the user to spend effort learning and becoming familiar with the technologies. Domestic robots are easier to use through their natural human-like communication, which can provide a pleasant experience for the user. Their friendliness can create an emotional bond that helps users, such as the elderly, feel more comfortable using them. Domestic robots are in fact effective informers, educators, reminders, and even readers of the users feelings and thoughts, which are hard to detect using other devices. While this effort is preliminary, and requires several technological advances, it is likely of significant interest for effective pervasive healthcare.

Multiple sensor mining technologies have been combined with such smart environment data gathering infrastructures to build healthcare applications targeting different requirements. The work in [81] uses frequent pattern mining to identify repeating structures in the routine patterns of human activity from environmental sensor data and detect changes in these patterns. This is important as the onset or complication of a life threatening episode may be marked by changes in behavior and activity patterns. This has been shown to be true for several conditions including prostatism, degenerative joint disease, bursitis, and gastro-

esophageal reflux, along with congestive heart failure, coronary artery disease, and chronic obstructive pulmonary disease.

Sensor mining, on data collected from a combination of body sensors and smart environments, has been used successfully for automatic assessment of ADL-IADL activities. In [102] RFID tags are attached to different key objects with which a person interacts for a specific set of activities. The data from these tags is augmented by accelerometers placed at diffrent strategic locations on the person (such as wrist, hip, and thigh). The combined dataset is analyzed using different feature extraction and mining and classification techniques. The computed features include statistical properties such mean, variance, energy, spectral entropy, pairwise correlation between the three axes, and the first ten FFT coefficients and exponential FFT bands, computed over sliding windows shifted in increments of 0.5 seconds. For classification of activities the authors use three different approaches, namely Naive Bayes, Hidden Markov Models (HMMs) and Joint Boosting. They show that Naive Bayes and HMM classifiers are well-suited for low-level activities such as sitting, standing and walking or wood workshop activities. The Joint Boosting method is successfully applied to overcome limitations of the sensing and feature extraction. The results show that combined recognition helps in cases when tagged objects are being shared among the activities, as well as in periods when the RFID reader can not detect interactions with objects due to its short range. The authors also consider extensions of this work to include techniques for accurate activity recognition with reduced supervision.

Researchers from the Imperial College [80] have developed an ear-based Activity Recognition (e-AR) sensor that identifies four different levels of activity ranging from almost no activity (during sleeping or sitting for example) to activities involving a lot of movement (running, exercising). The activity level is continuously detected using a classifier applied to the accelerometer measurements and streamed from the e-AR device every 4 seconds. While some activities may be described by a single activity level, many activities produce a sequence of activity levels. The work in [81] uses the output of the e-AR sensor to efficiently mine and update a concise variable-resolution synopsis routine for efficient behavior profiling in a home healthcare environment. The authors use the FP-Stream [82] and Closet+ [83] mining algorithms to describe behavior patterns using a routine tree data structure. The authors demonstrate that using this technique they can identify frequent patterns to describe the structure present in an individuals daily activity, and can then analyze both routine behavior as well as deviations.

4.3 Reality Mining

Reality mining [72] has recently been identified as one of 10 emerging technologies that could change the world, as it allows us to build comprehensive pictures of our lives, with the potential of transforming our understanding of ourselves, our organizations, and our society. Reality mining pulls together digital trace data that we generate as part of our daily activities with data mining and machine learning techniques to enable new non-intrusive applications in diagnosis, patient and treatment monitoring, health services use, surveillance of disease and risk factors, and public health investigation and disease control.

One of the key sensors employed by reality mining techniques is the mobile phone - that has become a central part of our lives. Mobile phones currently capture a lot of contextual information about users, including location (communication between the device and towers or GPS sensors) as well as data about their social connections (call and duration information). In addition, newer smart phones, e.g. the iPhone, include special sensors such as the microphone or the accelerometers that allow the capture of important diagnostic and health related data. These devices now also have the processing power of low-end desktop computers, allowing the deployment of several local analytics in support of healthcare applications.

Reality mining of these behavior signals may be correlated to the function of some major brain systems. It has been shown that arousal of the autonomic nervous system produces changes in activity levels. Hence, recent pilot projects have shown that it may be possible to diagnose depression from the way a person talks – depressed people tend to speak more slowly, a change that speech analysis software on a phone might recognize more readily than friends or family do [111]. Similarly, monitoring a phones motion sensors can also reveal small changes in gait, which could be an early indicator of ailments such as Parkinsons disease.

The phone sensors may be used to measure time-coupling between speech and movement of people, to capture indications of attention and screen for language development problems. The sensors can potentially capture the unconscious mimicry between people (e.g., reciprocated head nods, posture changes, etc.) as reliable predictors of trust and empathy, and improve compliance [112]. Similarly, the sensors can also be used to measure consistency or fluidity of movement or speech production to capture cognitive load. These different types of measurements of brain function have been shown to be predictive measures of human behavior [113], and play an important role in human social interactions

thereby supporting new methods of diagnosis, treatment monitoring, and population health assessments.

In addition to these automated measurement streams from the phone sensors, these devices may also be used to collect self-report data. Self-reported data from individuals during the course of their daily lives includes information such as symptoms, schedule, substance use, and mood that offer direct assessment of their cognitive and emotional states, perceptions of events, and information on the contexts in which they are involved. In many cases, the outcomes of interest in medicine and public health, such as some kinds of symptoms, can be measured only through self-report. By gathering self-reported data jointly with other reality mining data streams, errors can be reduced and dynamic aspects of health phenomena can be revealed.

Besides information on individual health, cell phones can be used to capture information about social relationships and social networks. Several pilot studies have shown how combined information on user location, proximity to other users, call and SMS patterns, and (with phones that have accelerometers) user motion can identify different patterns of behavior depending upon the social relationship between people. In [72] it has been shown that self-reported reciprocal friends (both persons report the other as a friend), non-reciprocal friends (only one of a pair reports the other as a friend), and reciprocal non-friends (neither of a pair reports the other as a friend) exhibit very different patterns. It has been shown that coupled with appropriate statistical analysis user social networks of friends and co-workers can be identified with average accuracies of up to 96 percent [114]. Such information has been shown to useful for several healthcare applications including reinforcing active learning. In [115] the authors describe DiaBetNet, a computer game for young diabetics that leverages smart phone functionality to encourage young diabetics to keep track of their food intake, activity, and blood sugar level.

Several government health services rely on demographic data to guide service delivery. Reality mining also provides a way to characterize behavior, and thus provides a classification framework that is more directly relevant to health outcomes [113]. Reality mining research has shown that most people have only a small repertoire of behavior patterns, and that this small set of behavior patterns accounts for the vast majority of an individual activity. Understanding the behavior patterns of different subpopulations and the mixing between them is critical to the delivery of public health services, because different subpopulations have different risk profiles and different attitudes about health-related choices. The use of reality mining to discover these behavior patterns can potentially

provide great improvements in health education efforts and behavioral interventions.

Other attempts to model large-scale population health include Google Flu Trends [116] to detect influenza outbreaks indirectly by tracking the frequency of World Wide Web searches for terms related to influenza-like illnesses. For geographic areas as small as states in the U.S., Google researchers have demonstrated that such search frequencies correlate strongly with estimated influenza incidence based on conventional surveillance of cases detected in a Centers for Disease Control and Prevention (CDC) network of sentinel laboratories and physicians. Similarly, the Automated Epidemiologic Geotemporal Integrated Surveillance System (AEGIS), developed by Children Hospital Boston, involves Internet-based data collection, management, and analysis systems to produce timely estimates of incidence. Almost 30,000 residents of Belgium, the Netherlands, and Portugal voluntarily report on their influenza symptoms on a weekly basis at the Gripenet web sites [117].

Reality mining can also significantly impact epidemiologic investigations that capture impact of exposure to different types of environments and pathogens on population health[7] . For instance, traditional investigations into links between individual exposures to airborne pollutants (particulate matter, carbon monoxide, and nitric oxide) and health conditions have relied on comparisons of aggregates of persons, or static measures and snapshots of exposure. This has impacted the effectiveness of such studies, and the associated costs. As opposed to these aggregate or static approaches, reality mining can be used to capture dynamic measures of time-activity patterns in relation to exposures. The cell phone location data can be combined with existing air quality monitoring stations or inferred from vehicle traffic patterns and locations of industrial facilities to yield spatially precise measures of exposure suitable for studying large samples of individuals.

While the discussion on reality mining in this chapter has been dominated by information captured from individual mobile phones, several aspects of our cities are getting instrumented. This includes our transportation infrastructures, security infrastructures, energy and utility systems, food production and distribution etc. Combining all of this information at scale, overcoming the associated data ownership, privacy, and connectivity challenges, and analyzing it can provide significant benefits

[7]The Spatio-Temporal Epidemiological Modeler (STEM) [118] activity tool has recently been proposed as an open source application designed to help scientists and public health officials create and use models of emerging infectious diseases.

towards improving the delivery and advancement of healthcare both for personal healthcare as well as population health management.

5. Summary and Concluding Remarks

This chapter surveys the application of sensor data mining in medical informatics. With the general increased instrumentation of the world with sensors, the need to make healthcare delivery more proactive, the ability to mine sensor data in healthcare is receiving a significant amount of attention. Despite these efforts, several challenges both technical and non technical remain to be solved. We have surveyed these challenges in this chapter, before presenting illustrative applications of sensor data mining technologies, both for clinical and non-clinical applications.

References

[1] "Health care - Definition from the Merriam-Webster Dictionary" http://www.merriam-webster.com/dictionary/health%5C%20care

[2] W. Stead, "Medical Informatics On the Path Toward Universal Truths," J Am Med Inform Assoc. 1998 Nov-Dec; 5(6): 583-584.

[3] V. L. Patel and D. R. Kaufman,"Science and Practice: A Case for Medical Informatics as a Local Science of Design," J Am Med Inform Assoc. 1998 Nov-Dec; 5(6): 489-492.

[4] "HIMSS Analytics Survey Demonstrates Awareness and Usage of Remote Health Monitoring Devices," retrieved from https://www.himssanalytics.org/about/NewsDetail.aspx?nid=79508

[5] "Instant Heart Rate," retrieved from http://www.azumio.com/apps/heart$-$rate/

[6] "Using the Nike+iPod Sensor," retrieved from http://walking.about.com/od/pedometer1/ss/nikeplussensor_4.htm

[7] J. Han and M. Kamber,"Data Mining: Concepts and Techniques, Second Edition," The Morgan Kaufmann Series in Data Management Systems, 2005.

[8] "CRISP DM 1.0, Step-by-step data mining guide," retrieved from http://www.the-modeling-agency.com/crisp-dm.pdf

[9] "Health Care Devices," retrieved from http://www.hl7.org/special/committees/healthcaredevices/overview.cfm

[10] "Continua Health Alliance," retrieved from http://www.continuaalliance.org/index.html

[11] C. Park, J. Lim and S. Park, "ISO/IEEE 11073 PHD standardization of legacy healthcare devices for home healthcare services," IEEE International Conference on Consumer Electronics (ICCE), 2011

[12] C. Aggarwal and P. Yu (eds), "Privacy Preserving Data Mining: Models and Algorithms," Springer, 2008

[13] R. Agrawal and R. Srikant, "Privacy Preserving Data Mining," SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000

[14] S. Sanei and J. A. Chambers, "EEG Signal Processing," Wiley-Interscience; 1st edition, 2007.

[15] J. Saunders, "The practice of clinical medicine as an art and as a science," Med Humanities 2000;26:18-22.

[16] N. Lee, A. F. Laine, J. Hu, F. Wang, J. Sun and S. Ebadollahi, "Mining electronic medical records to explore the linkage between healthcare resource utilization and disease severity in diabetic patients," First IEEE International Conference on Healthcare Informatics, Imaging and Systems Biology (HISB) 2011.

[17] A. Wright, T. N. Ricciardi and M. Zwick, "Application of Information-Theoretic Data Mining Techniques in a National Ambulatory Practice Outcomes Research Network," AMIA Annu Symp Proc. 2005; 2005: 829-833.

[18] T. Botsis, G. Hartvigsen, F. Chen and C. Weng, "Secondary Use of EHR: Data Quality Issues and Informatics Opportunities," AMIA Summits Transl Sci Proc. 2010; 2010: 1-5.

[19] I. Batal, H. Valizadegan, G. Cooper and M. Hauskrecht, "A Pattern Mining Approach for Classifying Multivariate Temporal Data," IEEE International Conference on Bioinformatics and Biomedicine , Atlanta, Georgia, November 2011.

[20] R. Trowbridge and S. Weingarten, "Clinical Decision Support Systems," retrieved from
`http://www.ahrq.gov/clinic/ptsafety/chap53.htm`

[21] N. Wickramasinghe, R. Bali, M. Gibbons, J. Choi and J. Schaffer, "A Systematic Approach Optimization of Healthcare Operations with Knowledge Management," JHIM Summer 2009, volume 23 / Number 3, www.himss.org.

[22] B. Brenn,"Using your EMR to Improve Operational Efficiency," retrieved from `http://www.pedsanesthesia.org/meetings/2009annual/syllabus/pdfs/submissions/Using\%20your\%20EMR\%20to\%20improve\%20operational\%20efficiency-B\%20Randall\%20Brenn\%20MD.pdf`

[23] C. Tsien and J. Fackler, "Poor prognosis for existing monitors in the intensive care unit," Crit Care Med, 1997 Apr, 25:4, 614-9

[24] A. Bar-Or, J. Healey, L. Kontothanassis and J. Van Thong, "BioStream: a system architecture for real-time processing of physiological signals," IEEE Engineering in Medicine and Biology Society, 2004. IEMBS 04. 26th Annual International Conference, 2004.

[25] H. Hyoil, R. Han, and H. Patrick. "An infrastructure of stream data mining, fusion and management of monitored patients," In IEEE Symposium on Computer Based Medical Systems, 2006.

[26] P. Norris and B. Dawant, "Knowledge-Based Systems for Intelligent Patient Monitoring and Management in Critical Care Environments," The Biomedical Engineering Handbook, Second Edition. 2 Volume Set Edited by Joseph D . Bronzino CRC Press 1999

[27] M. Blount, M. Ebling, M. Eklund, A. James, C. McGregor, N. Percival, K. Smith, and D. Sow, "Real-time analysis for intensive care: development and deployment of the artemis analytic system," IEEE Engineering in Medicine and Biology Magazine, Vol 29, Issue 2, pp. 110-118, May 2010.

[28] C. Chen, H. Agrawal, M. Cochinwala, and D. Rosenblut, "Stream query processing for healthcare bio-sensor applications" In 20th International Conference on Data Engineering, pages 791-794, 2004.

[29] D. Sow, M. Schmidt, D. Alberts, A. Beygelzimer, A. Biem, G. Luo and D. Turaga, "Developing and Deploying Clinical Models for the Early Detection of Clinical Complications in Neurological Intensive Care Units," 2011 AMIA Clinical Research Informatics Summit.

[30] J. Sun, D. Sow, J. Hu and S. Ebadollah,"A system for mining temporal physiological data streams for advanced prognostic decision support," In 10th IEEE International Conference on Data Mining, December 2010.

[31] S. Ebadollahi, J. Sun, D. Gotz, J. Hu, D. Sow and C. Neti, "Predicting Patient's Trajectory of Physiological Data using Temporal Trends in Similar Patients: A System for Near-Term Prognostics," 2010 AMIA Annual Symposium.

[32] "MIMIC (Multiparameter Intelligent Monitoring in Intensive Care) II Database [Online]',' available at `http://physionet.org/physiobank/database/mimic2db/`.

[33] L. Li, "Fast Algorithms for Mining Co-evolving Time Series," Ph.D. thesis, September 2011 CMU-CS-11-127

[34] N. Ramakrishnan, "Mining Electronic Health Records", IEEE Computer, 2010, Volume 43, Issue 10, Pages:77-81

[35] H. Neuvirth, M. Ozery-Flato, J. Hu, J. Laserson, M. Kohn, S. Ebadollahi and M. Rosen-Zvi, "Toward personalized care management of patients at risk: the diabetes case study" Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining Pages 395-403

[36] F. Wang, T. Syeda-Mahmood, V. Ercegovac, D. Beymer and E. Shekita, "Large-Scale Multimodal Mining for Healthcare with MapReduce," ACM IHI Conference 2010. FIXME

[37] "IBM Watson," retrieved from http://www-03.ibm.com/innovation/us/watson/index.html

[38] " IBM's Watson embarks on medical career," retrieved from http://www.computerworld.com/s/article/358871/IBM_s_Watson_to_Diagnose_Patients

[39] L. Mearian, "IBM's Watson expands cancer care resume," retrieved from http://www.computerworld.com/s/article/9225515/IBM_s_Watson_expands_cancer_care_resume

[40] "The Autonomic Nervous System," retrieved from http://www.ndrf.org/ans.html

[41] E. NeSmith, S. Weinrich, J. Andrews, R. Medeiros, M. Hawkins and M. Weinrich, "Systemic Inflammatory Response Syndrome Score and Race as Predictors of Length of Stay in the Intensive Care Unit," American Journal Of Critical Care, 2009, Volume 18, No. 4

[42] U. Jaffer, R. Wade and T. Gourlay, "Cytokines in the systemic inflammatory response syndrome: a review," Links 2009.

[43] S. Ahmad, A. Tejuja, K. Newman, R. Zarychanski and A. Seely, "Clinical review: a review and analysis of heart rate variability and the diagnosis and prognosis of infection," Crit Care. 2009;13(6):232. Epub 2009 Nov 24.

[44] J. Richman and R. Moorman, "Physiological time-series analysis using approximate, entropy and sample entropy," Am J Physiol Heart Circ Physiol, 2000;278:H2039-H2049

[45] D. Lake, J. Richman, P. Griffin and R. Moorman, "Sample entropy analysis of neonatal heart rate variability," Am J Physiol. 2002;283:R789-R797

[46] "The Society for Complexity and Acute Illness," retrieved from http://www.scai-med.org/

[47] T. Buchmnam, P. Stein and B. Goldstein, "Heart rate variability in critical illness and critical care," Current Opinion in Critical Care, August 2002 - Volume 8 - Issue 4 - pp 311-315

[48] S. Pincus and B. Singer, "Randomness and degrees of irregularity," Proc. Natl. Acad. Sci. USA Vol. 93, pp. 2083-2088, March 1996

[49] R. Bryce and B. Sprague, "Revisiting detrended fluctuation analysis," Sci. Rep. 2012/03/14/online Vol. 2

[50] M. Feder, N. Merhav, and M. Gutman, "Universal prediction of indi- vidual sequences," IEEE Trans. Inform. Theory, vol. 38, pp. 1258-1270, July 1992.

[51] C. Garrard, D. Kontoyannis and M Piepoli, "Spectral analysis of heart rate variability in the sepsis syndrome," Clinical autonomic research official journal of the Clinical Autonomic Research Society 1993, Volume: 3, Issue: 1, Pages: 5-13

[52] R. Winchell and D. Hoyt, "Spectral analysis of heart rate variability in the ICU: a measure of autonomic function," J Surg Res. 1996 Jun;63(1):11-6.

[53] H. Kennedy, "Heart rate variability - A potential, noninvasive prognostic index in the critically ill patient," Critical Care Medecine, 1998, Volume 26, Issue 2, Pages 213-214

[54] B. Goldstein, D. Fiser, M. Kelly, D. Mickelsen, U. Ruttimann and M. Pollack, "Decomplexification in critical illness and injury: relationship between heart rate variability, severity of illness, and outcome," Crit Care Med. 1998 Feb;26(2):352-7.

[55] W. Riordan, P. Norris, J. Jenkins and J. Morris, "Early Loss of Heart Rate Complexity Predicts Mortality Regardless of Mechanism, Anatomic Location, or Severity of Injury in 2178 Trauma Patients," Journal of Surgical Research 2009, Volume 156, Issue 2, Pages 283-289

[56] "Weaning And Variability Evaluation (WAVE)," retrieved from `http://clinicaltrials.gov/ct2/show/NCT01237886`

[57] N. Thakor and Y. Zhu, "Applications of adaptive filtering to ECG analysis: noise cancellation and arrhythmia detection," Biomedical Engineering, IEEE Transactions on, Aug. 1991 Vol. 38, Issue: 8

[58] V. Somers, M. Dyken, M. Clary and F. Abboud, "Sympathetic Neural Mechanisms in Obstructive Sleep Apnea," J. Clin. Invest. Vol. 96, October 1995, 1897-1904

[59] A. Shoeb. "Application of machine learning to epileptic seizure onset detection and treatment," PhD Thesis, Massachusetts Institute of Technology, 2009.

[60] B. Foreman and J. Claassen,"Quantitative EEG for the detection of brain ischemia," Critical Care 2012, 16:216

[61] H. Cao, L. Eshelman, N. Chbat, L. Nielsen, B. Gross and M. Saeed, "Predicting ICU hemodynamic instability using continuous multiparameter trends," IEEE Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference

[62] G. Rudedge, S. Andersen, J. Polaschek and L. Fagan, "A Belief Network Model for Interpretation of ICU Data," Proc Annu Symp Comput Appl Med Care ¿ Nov 7, 1990.

[63] J. Quinn and C. Williams, "Physiological Monitoring with Factorial Switching Linear Dynamical Systems," Chapter appearing in Bayesian Time Series Models, eds. D. Barber, A. Cemgil, S. Chiappa, Cambridge University Press, 2011.

[64] N. Aleks, S. Russell, M. Madden, D. Morabito, K. Staudenmayer, M. Cohen and G. Manley, "Probabilistic detection of short events, with application to critical care monitoring,", Neural Information Processing Systems (NIPS) 2008.

[65] F. Sharbrough, J. Messick and T. Sundt, "Correlation of continuous electroencephalograms with cerebral blood flow measurements during carotid endarterectomy," Stroke 1973, 4:674-683.

[66] O. Caelen, G. Bontempi, E. Coussaert, L. Barvais and F. Clement, "Machine Learning Techniques to Enable Closed-Loop Control in Anesthesia," 19th IEEE International Symposium on Computer-Based Medical Systems, 2006. CBMS 2006.

[67] S. Agarwal, A. Joshi, T. Finin, Y. Yesha and T. Ganous, "A pervasive computing system for the operating room of the future," Journal Mobile Networks and Applications archive, Volume 12 Issue 2-3, March 2007.

[68] C. Desouza, H. Salazar, B. Cheong, J. Murgo, and V. Fonseca, "Association of hypoglycemia and cardiac ischemia," Diabetes Care, 26(5):1485-1489, May 2003.

[69] M. Blount, V. Batra, A. Capella, M. Ebling, W. Jerome, S. Martin, M. Nidd, M. Niemi and S. Wright, "Remote health-care monitoring using Personal Care Connect," IBM Systems Journal Vol. 46 Issue 1, January 2007

[70] I. Mohomed, A. Misra, M. Ebling and W. Jerome, "HARMONI: Context-aware Filtering of Sensor Data for Continuous Remote Health Monitoring," Sixth Annual IEEE International Conference on Pervasive Computing and Communications (Percom 2008), Hong Kong, China, March 2008.

[71] Broan, Ian and Adams, Andrew, "The ethical challenges of ubiquitous healthcare," International Review of Information Ethics Vol. 8 (12/2007).

[72] A. Pentland, D. Lazer, D. Brewer and T. Heibeck, "Improving Public Health and Medicine by use of Reality Mining" Studies in Health Technology Informatics 2009;149:93-102.

[73] M. Chen, S. Gonzalez et al, "Body Area Networks: A Survey", Mobile Netw Appl (2011) 16:171-193.

[74] M. Garg, D. Kim, D. Turaga, B. Prabhakaran, "Multimodal Analysis of Body Sensor Network Data Streams for Real-time Healthcare," ACM MIR 2010.

[75] E. Seto, A. Giani, et al, A Wireless Body Sensor Network for the Prevention and Management of Asthma, in Proceedings of the IEEE Symposium on Industrial Embedded Systems (SIES), July, 2009.

[76] C. Seeger, A. Buchmann and K. Van Laerhoven, "myHealthAssistant: A Phone-based Body Sensor Network that Captures the Wearers Exercises throughout the Day," The 6th International Conference on Body Area Networks, 2011.

[77] D. Tate, R. Wing and R. Winett, "Using Internet Technology to Deliver a Behavioral Weight Loss Program," JAMA, 2001.

[78] S. Consolvo, D. McDonald, et al, "Activity Sensing in the Wild: A Field Trial of UbiFit Garden," Conference on Human Factors in Computing Systems, 2008.

[79] B. Tessendorf, A. Bulling, et al, "Recognition of Hearing Needs from Body and Eye Movements to Improve Hearing Instruments," International Conference on Pervasive Computing, 2011.

[80] B. Lo, L. Atallah, et al, "Real-Time Pervasive Monitoring for Postoperative Care," in Proc. 4th International Workshop on Wearable and Implantable Body Sensor Networks, Aachen, 2007, pp. 122 -127.

[81] R. Ali, M. Elhelw, L. Atallah, B. Lo and G. Yang,"Pattern Mining for Routine Behaviour Discovery in Pervasive Healthcare Environments," Proceedings of the 5th International Conference on Information Technology and Application in Biomedicine, May 2008.

[82] C. Giannella, J. Han, et al, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities, H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining, AAAI/MIT Press, 2004.

[83] J. Wang, J. Han and J. Pei, "CLOSET+: searching for the best strategies for mining frequent closed itemsets", in Proc. of the 9th ACM SIGKDD, 2003.

[84] O. Amft and G. Troster "Recognition of dietary activity events using on-body sensors," Artif Intell Med. 2008 Feb;42(2).

[85] J. Sriram, M. Shin, et al, "Activity-aware ECG-based patient authentication for remote health monitoring," ICMI-MLMI '09 Proceedings of the 2009 international conference on Multimodal interfaces .

[86] The Shimmer Platform, retrieved from `http://shimmer-research.com/`

[87] R. DeVaul, M. Sung, et al, "MIThril 2003: Applications and Architecture," International Symposium of Wearable Computers, October, 2003.

[88] M. Sung and A. Pentland, "LiveNet: Health and Lifestyle Networking Through Distributed Mobile Devices," Mobisys 2004.

[89] J. Weaver, "A Wearable Health Monitor to Aid Parkinson Disease Treatment," MIT M.S. Thesis, June 2003.

[90] M. Sung, "Shivering Motion/Hypothermia Classification for Wearable Soldier Health Monitoring Systems," Technical Report, MIT Media Lab, Dec. 2003

[91] "MyHeart - Fighting cardio-vascular diseases by prevention & early diagnosis," FP6 Integrated Project, `://www.hitechprojects.com/euprojects/myheart/`

[92] "Healthy Aims," `://www.healthyaims.org/`

[93] "Wearable Health Care System," `://www.wealthy-ist.com/`

[94] "Motion Analysis Lab," `://www.spauldingnetwork.org/research/motion-analysis-lab.aspx`

[95] T. Gao, C. Pesto, et al, "Wireless Medical Sensor Networks in Emergency Response: Implementation and Pilot Results," in Proceedings of the 2008 IEEE conference on Technologies for Homeland Security, pp:187-192, Waltham, MA, May 2008.

[96] M. Tomita, L. Russ, R. Sridhar and B. Naughton, "Smart home with healthcare technologies for community-dwelling older adults," Machiko R. Tomita, Linda S. Russ, Ramalingam Sridhar, Bruce J. Naughton M. (2010). Smart Home with Healthcare Technologies for Community-Dwelling Older Adults, Smart Home Systems, Mahmoud A. Al-Qutayri (Ed.), InTech.

[97] "Smart Medical Home research laboratory. University of Rochester," Retrieved from `://www.futurehealth.rochester.edu/smart;home/`

[98] "Georgia Institute of Technology (2009) Aware Home Research Institute," Retrieved from `://awarehome.imtc.gatech.edu`

[99] S. Helal, W. Mann, et al, "The Gator Tech Smart House: A programmable pervasive space," IEEE Computer, Vol. 38, No. 3, (March, 2005) 64-74.

[100] D. Cook and L. Holder, "Sensor selection to support practical use of health-monitoring smart environments," Wiley Interdisc. Rew., Data Mining and Knowledge Discovery, 2011.

[101] D. Lee, S. Helal, et al, "Participatory and Persuasive Telehealth," Gerontology, 2011.

[102] M. Stikic, T. Huynhy, K. Van Laerhoveny and B. Schieley,"ADL Recognition Based on the Combination of RFID and Accelerometer Sensing," Pervasive Computing Technologies for Healthcare, 2008.

[103] M. Hodges, N. Kirsch, M. Newman and M. Pollack,"Automatic Assessment of Cognitive Impairment Through Electronic Observation of Object Usage," Pervasive Computing 2010.

[104] H. Jimison, M. Pavel and J. McKanna, "Unobtrusive computer monitoring of sensory-motor function," Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference (September 2005).

[105] M. Borazio and K. Van Laerhoven, "Combining Wearable and Environmental Sensing into an Unobtrusive Tool for Long-Term Sleep Studies," IHI 2012.

[106] "PhMon Personal Health Monitoring System with Microsystem Sensor Technology," retrieved from `http://www.phmon.de/englisch/index.html`

[107] V. Jones, A. Van Halteren, et al, "MobiHealth: Mobile Health Services based on Body Area Networks," M-Health Emerging Mobile Health Systems. Springer-Verlag, Berlin, pp. 219-236.

[108] J. Rodriguez, A. Goni and A. Illarramendi, "Real-time classification of ECGs on a PDA," Information Technology in Biomedicine, IEEE Transactions on, Volume 9, Issue 1, March 2005 Page(s): 23-34.

[109] P. Leijdekkers and V. Gay, "Personal Heart Monitoring and Rehabilitation System using Smart Phones," Mobile Business, 2006. ICMB '06.

[110] L. Zeng, P. Hsueh and H. Chang,"Greenolive: an Open Platform for Wellness Management Ecosystem," Service Operations and Logistics and Informatics (SOLI), 2010

[111] W. Stoltzman,"Toward a social signaling framework: Activity and emphasis in speech," Master thesis, MIT EECS, 2006.

[112] J. Bailenson and N. Yee,"Digital chameleons: Automatic assimilation of nonverbal gestures in immersive virtual environments," Psychological Science, 16(10): 814-819, 2005.

[113] A. Pentland,"Honest signals: How they shape your world," MIT Press, 2008.

[114] W. Dong and A. Pentland," Modeling influence between experts," Lecture Notes on AI: Special Volume on Human Computing, 4451: 170-189.

[115] V. Kumar and A. Pentland, "DiaBetNet: Learning and predicting blood glucose results to optimize glycemic control," 4th Ann. Diabetes Technology Meeting, Atlanta, GA.

[116] J. Ginsberg, M. Mohebbi, et al, "Detecting influenza epidemics using search engine query data," Nature (in press).

[117] S. Van Noort, M. Muehlen, et al, "Gripenet: An internet-based system to monitor influenza-like illness uniformly across Europe," Eurosurveillance, 12(7): 2007. `http://www.eurosurveillance.org/ViewArticle.aspx?ArticleId=722`.

[118] "Spatio Temporal Epidemiological Modeler," `http://www.almaden.ibm.com/cs/projects/stem/`.

Chapter 15

EARTH SCIENCE APPLICATIONS OF SENSOR DATA

Anuj Karpatne, James Faghmous, Jaya Kawale, Luke Styles, Mace Blank, Varun Mithal, Xi Chen, Ankush Khandelwal, Shyam Boriah, Karsten Steinhaeuser, Michael Steinbach, Vipin Kumar

Department of Computer Science and Engineering
University of Minnesota
{anuj,faghmous,kawale,styles,blank,mithal,chen,ankush,sboriah,ksteinha,steinbac,kumar}@cs.umn.edu

Stefan Liess

Department of Soil, Water and Climate
University of Minnesota
liess@umn.edu

Abstract Advances in earth observation technologies have led to the acquisition of vast volumes of accurate, timely and reliable environmental data which encompass a multitude of information about the land, ocean and atmosphere of the planet. Earth science sensor datasets capture multiple facets of information about natural processes and human activities that shape the physical landscape and environmental quality of our planet, and thus, offer an opportunity to monitor and understand the diverse phenomena affecting earth's complex system. The monitoring, analysis and understanding of these rich sensor datasets is thus of prime importance for the efficient planning and management of critical resources, since the societal costs of mitigation or adaptation decisions for natural or human-induced adverse events are significant. Hence, a thorough understanding of earth science sensor datasets has a direct impact on a range of societally relevant issues. Moreover, earth science sensor datasets possess unique domain-specific properties that distinguish them from sensor datasets used in other domains, and thus demand the need for novel tools and techniques to be developed for their analysis, adhering to their characteristic issues and challenges.

Keywords: Remote Sensing, Earth Science, Data Mining

1. Introduction

Climate and earth sciences have recently experienced a rapid transformation from a data-poor to a data-rich environment. With the recent advances in earth monitoring technologies, observations from remote sensors on satellites and weather radars, or from *in situ* sensors and sensor networks, as well as outputs of climate or earth system models from large-scale computational platforms, provide terabytes of temporal, spatial and spatio-temporal data about earth's complex processes. In addition, the increasing use of geographical information systems for decision-making has provided an additional source of large spatial datasets. These massive and information rich datasets offer a huge potential for advancing earth science research and its impacts on related domains. Examples of earth science research tasks include developing robust, global-scale algorithms that provide spatially explicit and regularly updated techniques for global monitoring of the entire earth's land surface and oceans, determining relationships between multivariate events and variables, etc.

The understanding and monitoring of earth science sensor datasets offer unique data-centric and algorithmic challenges imposed by the volume, variety and richness. It is not only the massive size of datasets that poses a challenge, but also complexities due to the unique data characteristics such as spatial heterogeneity, temporal variability, and uncertainty. An additional challenge arises from the broad range of questions posed by diverse scientific disciplines covered in the broad purview of ecosystem or environmental sciences. Specifically, the analysis and discovery approaches need to be cognizant of climate and ecosystem data characteristics, the value of physically-motivated conceptual understanding and functional associations of the earth's system, as well as possible thresholds and tipping points in the impacted natural, engineered, or human systems. Thus, there is a strong need for understanding and advancing the state of the art in computational algorithms and robust data analysis methods which are tailored for applications in the earth science domain, crossing the traditional boundaries between computer science and earth science.

The analysis and discovery techniques for understanding and monitoring earth science sensor datasets can be broadly classified into the following two categories - (i) event detection, and (ii) relationship mining. First, detecting events of interest over land, ocean and atmosphere using multiple data sources enables earth scientists to monitor natural as well as anthropogenic processes and to quantitatively assess their environmental and socioeconomic impact. Second, finding relationships between spatio-temporal events and variables is crucial for improved understand-

ing of the interactions between different processes of the earth system at large, and thereby improving predictive power. Together, these two research tasks offer fertile grounds for developing novel knowledge discovery approaches which focus on addressing a range of interconnected, societally-relevant themes at the core of impending environmental concerns cutting across diverse disciplines. They enable a wide community to analyze changes in the earth's system, interactions between different processes from local to global scales, and their impacts on the carbon cycle, hydrology, air quality, biodiversity, and other research areas. In particular, qualitative inferences about changes and relationships in the earth's system and their impacts may be transformed into quantitative historic and predictive insights based on a combination of hypothesis-driven and data-guided discovery processes.

The remainder of the chapter is organized as follows. In Section 2, we provide an overview of the types of sensor datasets used in earth science research. Section 3 focuses on the data-centric challenges posed by earth science applications. Sections 5 and 6 introduce two broad categories of research problems using earth science data, namely event detection and relationship mining, providing illustrative examples in each category. Section 7 contains concluding remarks and directions for future work.

2. Overview of Earth Science Sensor Datasets

Earth science sensor datasets possess varying data characteristics, acquisition methods and domains of coverage (both in space and time). They either consist of local sensor recordings (*in situ* data) or are obtained through instruments mounted on satellites or other remotely based locations (remote sensing data). *In situ* sensors which are non-uniformly distributed in space at local or regional scales can be processed and made available at a fixed spatial grid using basic interpolation, aggregation and sampling techniques so that the processed data is free from missing values or non-uniformly spaced data. Further, interpolation methods can range from simple linear interpolation to reanalysis techniques using climate simulation models. We provide a brief overview of the diverse types of datasets used in earth science research in the following subsections.

2.1 Observational Data

Observational datasets that are commonly used in earth science research can be broadly classified into *station-based* and *gridded* data.

Station-based data. Due to the large-scale nature of earth's observed systems, earth science datasets are rarely directly measured on a regular coordinate system, except for small-scale experiments that resemble laboratory studies. Sensor observations in earth science research are generally obtained from irregularly and non-uniformly spaced stations such as weather stations over land, on ships, ocean buoys, and balloon measurements over a vertical section of the atmosphere [57, 67]. They provide the most direct and therefore least error-prone data sources available, since they do not employ complex post-processing operations for handling missing values or non-uniformity present in the data. However, *in situ* sensors are limited in their spatial and temporal coverage limiting their application to local or regional scale analyses only. Further, *in situ* sensors generally suffer from other physical limitations which introduce additional errors and uncertainty in these datasets. For example, balloon measurements do not strictly represent a vertical section of the atmosphere, but weather balloons rather are free to move in the horizontal direction during their ascent, making the analysis of such datasets difficult. Hence, in addition to *in situ* measurements, remote sensors such as ground-based radar imagery and satellite instruments are able to measure diverse properties about the earth over large distances. Since polar-orbiting satellites revolve around the planet, remote sensors cannot provide continuous measurements over specific locations.

Gridded data. Most earth science sensor recordings are post-processed using basic interpolation, aggregation and sampling techniques to provide easily accessible datasets at a fixed spatial grid, with a particular spatial resolution, and available at regular time intervals. Figure 15.1 provides a schematic representation of multiple grid-based datasets such as Sea Surface Temperature (SST), Precipitation, Pressure, and Net Primary Productivity (NPP). As an example, station-based surface temperature and precipitation measurements are interpolated to two-dimensional horizontal grids (spatial resolution ranging from 0.5 degrees to 5 degrees) [19], and satellite data usually undergo a number of steps before being released, e.g. calibration, orbital correction, quality control, and conversion to regular grids [1]. Satellites have successfully monitored a number of attributes about the earth such as surface temperature, humidity, clouds and chemical composition of the atmosphere. However, certain details in atmospheric thermodynamics and dynamics need to be resolved by *in situ* measurements, specially when sensor observations are obtained from spatially distant remote sensor stations, making it difficult for simple interpolation algorithms to function.

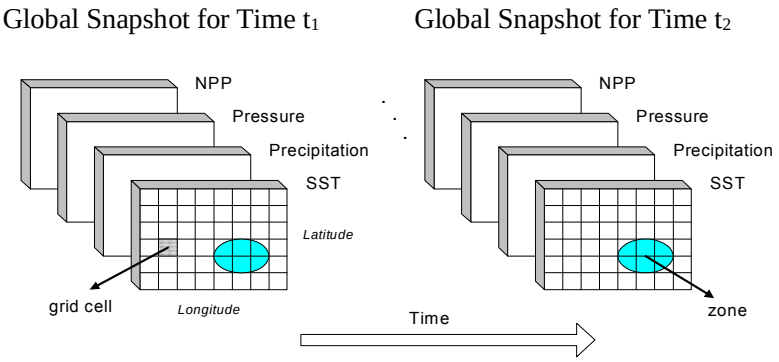


Figure 15.1. Schematic of multiple grid-based remote sensing datasets such as Sea Surface Temperature (SST), Precipitation, Pressure, and Net Primary Productivity (NPP), represented on a spatial grid, where each grid cell covers a range of latitude and longitude values, and varying with time

2.2 Reanalysis Data

When observational sensor datasets are scantily available or are irregularly placed rendering simple interpolation methods functionless, comprehensive physical models have to be used in conjunction with observed sensor recordings to calculate possible values over large areas with missing values or data of poor quality. The analysis and interpolation of observed sensor recordings requires physical knowledge of involved natural processes to fill in missing or poor quality values. For atmospheric and oceanic data, this knowledge is provided by general circulation models [52], e.g. the Goddard Earth Observing System Data Assimilation System Version 5 (GEOS-5) [33]. After generating analysis fields at each time by assimilating observations into the physical model, the model verifies the consistency of the data products over time and makes appropriate adjustments to the data by balancing between observational uncertainty and acceptable noise in the system. This step is referred to as *reanalysis*. Multiple *reanalysis* products are currently available, each with different input sources and underlying physical model [64].

Table 15.1 lists commonly used datasets in earth science research. Since earth science datasets exhibit unique properties distinguishing them from sensor datasets used in other domains, we discuss key data-centric issues and challenges in analyzing earth science datasets in the subsequent section.

Data Type	Source(s)	Use
Spectral Reflectance	Centre National d'Etudes Spatiales[59], NASA[55], NOAA[7], USGS[71]	Spectral reflectance is used to compute vegetation indices, surface temperature and a number of other variables. These variables are fundamental to studies in forestry, agriculture and urbanization.
River Discharge	German Federal Institute of Hydrology[30], SAGE[70]	River discharge levels are an important component of the hydrological cycle, which is in turn connected to agriculture and urbanization.
Nighttime Lights	Department of Defense, NOAA	Mapping urbanization dynamics.
Aerosols	NASA[56], World Data Centre for Aerosols[78], Japan Aerospace Exploration Agency[38]	Atmospheric aerosol concentration is often higher in urban areas. Aerosol concentration impacts regions temperature and precipitation patterns.
Carbon Cycle Greenhouse Gases	NOAA[58]	Impacts of land disturbances on the carbon cycle.
Digital Elevation Model	METI[49], USGS[72]	Topography can affect landslide risk, spread of wildfires, agricultural productivity, potential for urbanization, etc.
Climate Data	NCDC[57], SPARC[67]	Climate datasets that are obtained through *in situ* or satellite based sensors consist of information about the land, ocean and atmosphere such as temperature, pressure, sea surface height, precipitation etc.
Model Data	MERRA[52], GEOS5[33], RIO[64]	Model data generally comprise of reanalysis datasets that supplement observational data with physics based models, such as general circulation models, for interpolating and projecting over regions and time intervals where sensor observations are sparse.

Table 15.1. Sensor datasets used in earth science research.

3. Data-centric Challenges

Earth science datasets pose several unique challenges: some are due to their inherent spatial-temporal nature and others are specific to the domain. The following paragraphs throw some light on the key characteristics and challenges of sensor datasets used in earth science research.

Spatial-temporal data. Many earth science datasets have been interpolated to gridded time series for ecosystem and environmental variables, i.e., each series represents an individual co-registered cell in a latitude-longitude grid that covers the entire surface (or a region) of the Earth.

Uncertainty and incompleteness. Earth science datasets are frequently plagued with noise/uncertainty and incompleteness due to sensor interference and instrument malfunctions. This issue is particularly acute in the case of remotely sensed land surface data, where atmospheric (clouds and other aerosols) and surface (snow and ice) interference are constantly encountered. This motivates the need for development of algorithms that are robust to presence of uncertainty and incompleteness in data.

Temporal variability. Ecosystem observations tend to have a high degree of temporal variation. For example, vegetation data such as greenness usually changes naturally on multi-year scale, but infrequent and local events such as forest fires and logging can induce short-time events in naturally occurring spatio-temporal processes. These events need to be distinguished from other more regularly occurring events such as the seasonal cycle and recurring rain seasons. Handling such naturally occurring temporal variations is necessary to avoid detection of spurious patterns.

Spatial heterogeneity. This refers to variability of the observed processes over space and is illustrated by natural boundaries of wildfires or deforestation due to topographical constraints, growth of cities along a spatial gradient, or preferential land conversion for agricultural intensification near resources such as lakes and cities. Further, data heterogeneity drives the need for developing local or regional models, each corresponding to a homogeneous group of locations, into the data mining framework.

Multi-resolution and multi-scale. Naturally occurring global phenomena occur at different scales. For example, events such as urbanization, fires and deforestation tend to impact smaller areas than droughts. The degree of spatial heterogeneity of each dataset determines the necessary grid size to resolve important characteristics. Some datasets, e.g., population and political borders (important to connect events to political decision making) are usually available for predefined regions and need to be interpolated to the gridded space. Weighted average values can attributed to grid cells that cover multiple spatial regions. One common approach is to build a bridge between these disparate scales and develop algorithms that can identify patterns at multiple resolutions without upsampling all data to the highest resolution.

Spatial autocorrelation. Tobler's first law of geography states that "Everything is related to everything else, but near things are more related than distant things" [69]. Thus, the spatial dependence of earth science data needs to be incorporated into data mining algorithms.

In the following two sections, we discuss two broad applications of earth science sensor datasets - (i) event detection, and (ii) relationship mining. We further supplement the discussion by providing illustrative examples of problems and methods developed in each of the aforementioned earth science applications.

4. Event Detection

Identifying different kinds of occurrences of ecosystem events such as forest disturbances, agricultural intensifications, urban expansions, ocean eddies and high aerosol concentrations can provide earth scientists and policy-makers with timely information to mitigate and adapt to critical environmental pressures. Event detection aims at detecting anomalous and/or change behavior across multiple spatio-temporal variables spanning multiple facets of information about the earth. Spatio-temporal datasets such as vegetation indices, night-time lights, sea surface height, land surface temperature, precipitation, aerosol concentration, and population can be used for identification of these events, as they often exhibit a change or a characteristic pattern in one or more of these types of data.

Since different sensor datasets capture unique (and often complementary) information about events of interest, we can leverage multiple sources of information in earth science domain to (i) detect and characterize events that exhibit different event characteristics in different variables (ii) improve confidence in the significance of a detected event and

(iii) summarize and cluster events that exhibit similar spatio-temporal properties among multiple variables.

Applications of event detection techniques involve monitoring changes occurring over either land, ocean, atmosphere or biosphere. For instance, vegetation time series (e.g., the Enhanced Vegetation Index from the MODIS instrument aboard NASA's Earth observing Terra and Aqua satellites) can be used to detect a variety of events, such as deforestation, floods, fires, etc., that result in a perceptible change in vegetation [2, 50]. Figure 15.2 provides an example of a vegetation time series at a particular location which got burned in the year 2008, showing a significant drop in the vegetation value in that year. Furthermore, land cover change detection using vegetation time series can be further enhanced by utilizing information about the thermal anomaly time series (available through MODIS), for characterizing fire events which register a thermal anomaly observation, from other land cover events such as deforestation, droughts etc. As another example, sea surface height can be used for detecting ocean eddies which are swirls of ocean currents playing a crucial role in transporting water, salt, heat, and nutrients in the ocean, as well as driving the ocean's dynamics [23].

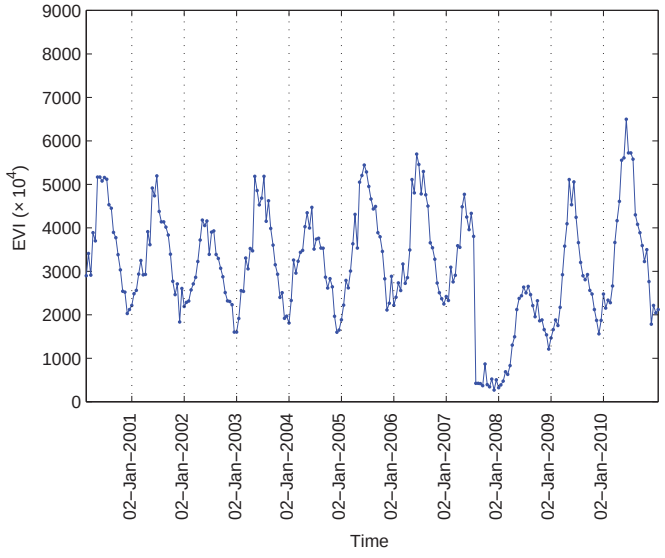


Figure 15.2. An example vegetation time series of a fire event at a particular location. The Enhanced Vegetation Index (EVI) shows a characteristic drop in the year 2008 during the event of the fire.

In the remainder of the section, we present two illustrative examples of change detection using remote sensing datasets - (i) land cover change monitoring, and (ii) identifying ocean eddy dynamics.

4.1 Illustrative Application: Monitoring Changes in Land Cover

Detecting meaningful events from global-scale earth science datasets poses several unique challenges which are yet to be addressed by traditional event detection techniques in the domain. Approaches that utilize only the spatial information using image snapshots disregard the rich temporal context of the data and are significantly impacted by temporal variability and noise. On the other hand, time series change detection methods do not utilize spatial information and thus are limited in their global applicability. Earth science datasets show varying characteristics in different climatic conditions and land cover types and are thus spatially heterogeneous. At the same time, locations that are spatially close to each other exhibit similar temporal variability due to spatial autocorrelation. This property influences the creation of a background model of unchanged locations in varying spatial regions. Furthermore, earth science datasets show similar values during the same seasonal duration across different years, due to the annual cycles of the earth. Additionally, similar event occurrences show spatial autocorrelation, which can be leveraged for improved event detection of spatially coherent events with low-intensity of change at varying degrees of representation.

Research on land cover change detection falls in three major categories namely spatial change detection, temporal change detection and spatio-temporal change detection. In the spatial domain, change detection has been framed as a classification problem and various approaches such as Markov random field (MRF) based methods [39], Gibbs-MRF [65] and Gaussian-MRF [80] have been developed. In addition, image comparison-based approaches (that compare snapshots of a region from different time steps usually separated by multiple years) have been developed in the earth science community and used to identify events such as urbanization, forest disturbances and agriculture related changes [20]. A major limitation of these methods is that they are inherently region-specific as high-quality training data is expensive to generate, and accuracy is poor if a classifier learned from the training samples of one geographical region is used for classifying test samples from another region. Furthermore, these methods fail to exploit the rich information in the temporal context. In particular, these methods are unable to identify the exact change duration, the rate of change and other use-

ful parameters related to the event that can be discovered by analyzing spatial-temporal data.

Time series change detection approaches can be broadly categorized under parameter change detection [15, 37, 68], anomaly detection [10, 36, 21, 63], prediction based methods [24, 73, 44, 40], and segmentation based approaches [34, 47, 46, 2]. Parameter change detection approaches assume that the detected event will exhibit a change in a characteristic parameter such as the mean or the variance of the data, and hence they can be identified by monitoring changes in the distribution of this statistic. They have been used in the past for mapping forest fires in Portugal as an example [45]. Anomaly detection approaches find subsequences that are unusual with the underlying hypothesis that observations deviate from normal when an event occurs [62]. Anomaly detection based on discord discovery finds subsequences that are significantly dissimilar to all other subsequences of the time series [18, 43]. Prediction based approaches explicitly learn a model that predicts future observations based on previous values, and deviation of the observed data from the model prediction is used as an indicator of change [31, 51]. Segmentation based approaches divide a time series into homogeneous parts (that can be approximated by a simpler generative process), where segments have high intra-segment similarity but low inter-segment similarity [28, 74, 2].

In the spatio-temporal context, event detection has been studied in the domain of sensor networks such as in [79, 77]. In the earth science domain, there is limited work that utilizes both spatial and temporal components effectively. Examples include [22] which characterize spatio-temporal fire activity patterns using satellite imagery, where airborne images are used for spatiotemporal change detection in forest cover [61], and classification-based approaches [9, 16].

A series of algorithms for land cover change detection [50] have recently been developed using predictive [51], segmentation [3, 4, 28] and parameter change [8] approaches on vegetation based time series data. These highly scalable algorithms overcome many of the limitations of traditional approaches to land cover change detection including the susceptibility to noise and temporal variability. The algorithms have been comparatively evaluated with state of the art land cover change detection techniques, and applied to global vegetation data (EVI) to detect a variety of changes in the global ecosystem including those due to fires, deforestation, insect damage, floods, hurricanes, conversion to agriculture, urbanization [50, 29, 51, 28, 8, 4, 5, 2].

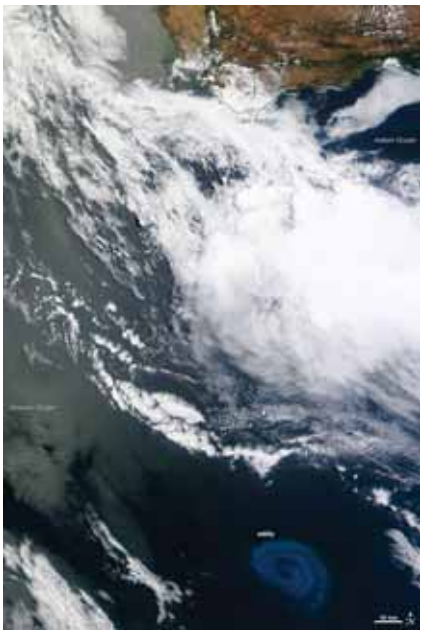


Figure 15.3. Image from the NASA TERRA satellite showing an anti-cyclonic (counter-clockwise in the Southern Hemisphere) eddy that likely peeled off from the Agulhas Current, which flows along the southeastern coast of Africa and around the tip of South Africa. This eddy (roughly 200 km wide) is an example of eddies transporting warm, salty water from the Indian Ocean to the South Atlantic. We are able to see the eddy, which is submerged *under* the surface because of the enhanced phytoplankton activity (reflected in the bright blue color). This anti-cyclonic eddy would cause a depression in subsurface density surfaces in sea surface height (SSH) data. Image courtesy of the NASA Earth Observatory.

4.2 Illustrative Application: Identifying Ocean Eddies from Satellite Altimeter Data

Coherent rotating structures of water, known as ocean eddies, are a crucial component of ocean circulation. In addition to dominating the ocean's kinetic energy, eddies play a significant role in the transport of water, salt, heat, and nutrients. Therefore, understanding current and future eddy patterns is a central challenge to addressing the sustainability of marine ecosystems.

Our understanding of ocean eddy dynamics has grown significantly with the advent of satellite altimetry. Prior to then, oceanographers relied primarily on case studies using drifting floats in the open ocean to collect detailed information about individual eddies such as rotational speeds, amplitude, and salinity profiles. With the increased accessibility

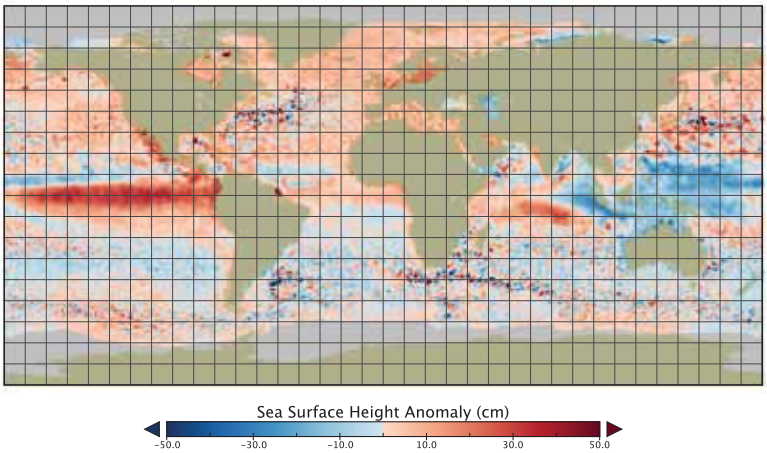


Figure 15.4. Global sea surface height (SSH) anomaly for the week of October 10 1997 from the Version 3 dataset of the Archiving, Validation, and Interpretation of Satellite Oceanographic (AVISO) dataset. Eddies can be observed globally as closed contoured negative (dark blue; for cyclonic) or positive (dark red; for anti-cyclonic) anomalies. Best seen in color.

to satellite data, ocean surface temperatures and color have been used to identify ocean eddies based on their signatures on such fields. While, these fields are impacted by eddy activity, there are additional phenomena that affect them as well effectively complicating eddy identification in such fields. More recently, sea surface height (SSH) observations from satellite radar altimeters have emerged as a better-suited alternative for studying eddy dynamics on a global scale given SSH's intimate connection to ocean eddy activity. Eddies are generally classified as either cyclonic if they rotate counter-clockwise (in the Northern Hemisphere) or anticyclonic otherwise. Cyclonic eddies cause a decrease in SSH and elevations in subsurface density surfaces. Anti-cyclonic eddies cause an increase in SSH and depressions in subsurface density surfaces. These characteristics allow us to identify ocean eddies in SSH satellite data. In Figure 15.4, anti-cyclonic eddies can be seen in patches of positive (dark red) SSH anomalies, while cyclonic eddies are reflected in closed contoured negative (dark blue) SSH anomalies.

Eddies can manifest themselves as local minima (maxima) embedded in a large-scale background of negative (positive) anomalies [13] (see Figure 15.5). Given the large variations in SSH at a global scale, tracking eddies globally presents several characteristic challenges. First, SSH data is prone to noise and uncertainty, making it difficult for meaningful eddy patterns to be distinguished from spurious events and noise.

Second, although eddies generally have an ellipse-like shape, the shape's manifestation in gridded SSH data differs based on latitude. This is because of the stretch deformation of projecting spherical coordinates into a two-dimensional plane. As a result, one cannot restrict eddies by shape (*e.g.* circle, ellipse, *etc.*) Finally, eddy heights and sizes vary by latitude, which makes having a global "acceptable" eddy size unfeasible [26]. Therefore, applying a single global threshold would wipe out many relevant patterns in the presence of spatial heterogeneity.

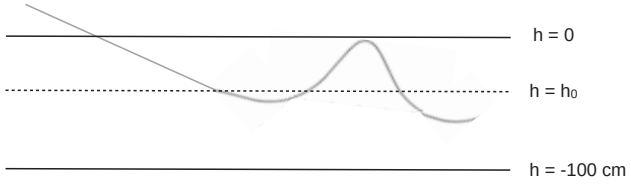


Figure 15.5. Schematic of an anti-cyclonic eddy that is embedded in a large scale background with a larger amplitude than the eddy. If we were to apply a threshold at $h = 0$ the eddy would be missed. This is motivation to use multiple threshold from $h = -100cm$ to $100cm$ as suggested by CH11. Figure adapted from [12]

Two major approaches have been used to monitor eddies globally: the first is spatial and the other temporal. In the spatial approach, eddies are identified as close-contoured positive or negative anomalies using classical connected component algorithms. These connected component algorithms tend to be highly parameterized to encode expert knowledge such as minimal eddy radius or amplitude to reduce the number of false positives. The state-of-the-art connected component approach was, however, computationally prohibitive and unable to separate eddies that were in close proximity. Using the insights provided by [26, 14], a recent study was able to address both shortcomings by applying a latitude dependent convexity criterion that is both efficient and accurate compared to the state-of-the-art methods. In the temporal domain, Faghmous et al. (2012) leveraged the spatio-temporal signature of ocean eddies on SSH to develop an unsupervised learning algorithm that identified groups of pixels that exhibited an eddy-like signature of slow decreases or increases in SSH over significant time periods. The temporal approach is significantly more efficient and robust than existing spatial methods alone - its computational complexity is linear in the resolution of the data compared to quadratic for the spatial approach, and given that only groups of pixels exhibiting similar eddy behavior are labeled as eddies, the temporal approach is more robust to outliers than the spatial one. This work, highlights the need of novel spatio-temporal data

mining approaches that are able to monitor features in continuous data fields, especially with the continual expansion of spatio-temporal climate datasets.

5. Relationship Mining

Predicting changes in the earth's system requires a comprehensive understanding of the complex feedback interactions among its underlying processes. Relationship mining aims at discovering information about the interactions between sensor attributes to provide an insight into the structure of the underlying phenomena. Instead of discovering anomalous or change events in all or some of the multiple variables, the primary objective here is to get a better understanding of the relationships between the variables in order to determine the structural properties of multivariate data. As an example, increased sea surface temperatures are known to affect the amount of precipitation received during the wet season in South America, eventually making the vegetation more susceptible to fire [17, 25]. In a recent work by Chen et al. [17], relationships between Sea Surface Temperature (SST) Anomalies in the Atlantic and Pacific Ocean, and Fire Season Severity (FSS) in South America were studied.

Through economic and policy actions, human behavior also often enters into these feedback structures. For example, deforestation for palm-oil plantation in peat-land regions of Indonesia is often followed by fires, in close spatial and temporal proximity [60]; such patterns are attributed to complex physical processes whereby deforestation leads to soil degradation, which reduces the moisture in the soil, making the peat reserves underneath more susceptible to fire leading to massive emissions of carbon [48]. As another example, increase in the average night-time winter temperature attributed to global warming in the higher latitudes of North America has led to an increase in pine beetle infestation in those regions. However attributing the effect of these infestations on forest fire frequency is under dispute as there have been studies supporting the effect [32] and otherwise [66] (see Figure 15.6).

As another example, relationships between events occurring at multiple locations can also be expressed as spatio-temporal patterns, evolving in space and time. Spatio-temporal sequential pattern mining aims at finding patterns of events that occur in close proximity of space and time. For example, Huang et al. [35] proposed a model for discovering sequential chains of events by extending the spatial co-location framework to spatio-temporal databases. Recently, Mohan et al. [53, 54] proposed a directed acyclic graph based approach to detect sequences of

events that appear as a cascade, capturing the partial orderedness of the event relationships. Another class of approaches have focussed on finding spatio-temporal patterns in a database of moving object trajectories (e.g., hurricane tracks and mobile users). In [75, 76], a spatio-temporal association rule (STAR) mining technique was proposed to capture the frequent appearance of moving objects at varying locations and time.

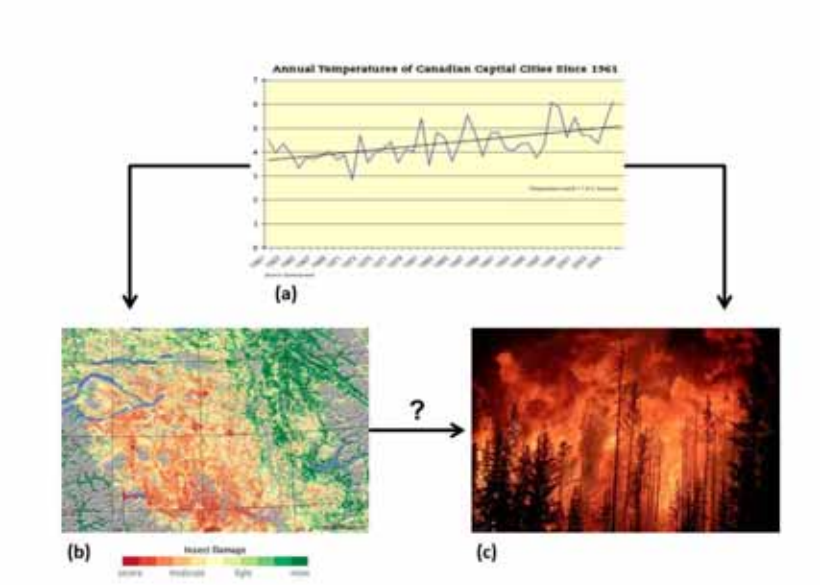


Figure 15.6. (a) Rise in the average winter temperatures of British Columbia, Canada; (b) Increase in pine beetle infestation (shown as points in red); (c) Occurrence of forest fires at these locations. Relationship between pine beetle infestation and forest fire events is unresolved [32, 66] and needs further study (Source: NASA & Environment Canada).

The ability to automatically extract such complex relationships from global-scale spatio-temporal data is essential to advance our current understanding of changes that can be attributed to natural and human-induced forcings and in turn advance our knowledge about global ecosystem dynamics at large. We next present an illustrative example of relationship mining for detecting teleconnections in climate data such as climate dipoles [41, 42], which are pairs of spatially distant locations exhibiting a relationship in their climate anomalies.

5.1 Illustrative Application: Identifying Atmospheric Teleconnections

Teleconnections are recurring patterns in climate anomalies connecting two spatial regions that are far apart from each other. They have been a subject of interest to climatologists due to the possibility of linking changes in weather at one location to changes at another distant location. Perhaps the most well known and widely studied teleconnection is the El Niño Southern Oscillation (ENSO). It represents a well defined "sea-saw" spatial pattern of surface pressure in the tropics and the subtropics exhibiting an oscillation. Figure 15.8 shows the sea level pressure anomaly (anomalies are constructed by removing the monthly means from the data and are widely used in climate to reduce the effects of seasonality) time series at Tahiti and Darwin which define the two ends of the Southern Oscillation and exhibit negative correlation. The El Niño climate phenomenon is known to be responsible for precipitation and temperature anomalies worldwide.

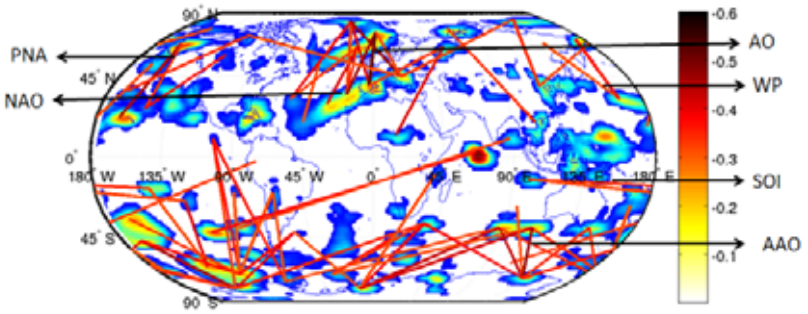


Figure 15.7. Map showing all the dipole edges in the NCEP Reanalysis dataset. Best seen in color.

Scientists have been cognizant of the existence of a number of teleconnections and historically they have been discovered by human observation or by using pattern analysis techniques such as the Empirical Orthogonal Function (EOF) over a limited region. However there are several limitations of the existing methods for finding these relationships, and they require considerable research and insight on the part of the domain experts involved. Knowledge of these teleconnections and their interactions is particularly important for predicting climate extreme events. For example, while the cold winter over Europe in 2010 could be largely explained by the North Atlantic Oscillation (NAO) which is another teleconnection, and other local indices, the cold winter

over North America at the same time is largely due to a combination of NAO and ENSO [27]. Further, the ability to address important questions like the degree of climate change and its potential impacts requires a deeper understanding of the behavior and interactions of these atmospheric processes as well as to capture them precisely. Discovery of relationships or dependencies among climate variables involved is extremely challenging due to the nature and massive size of the data. Data-guided approaches thus offer a huge potential for characterizing and discovering unknown relationships along with advancing climate science.

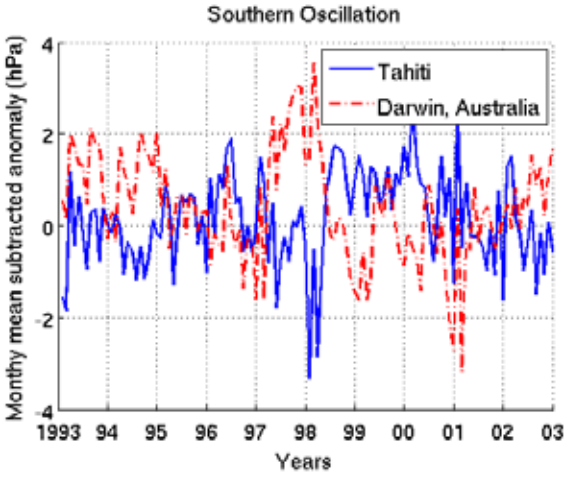


Figure 15.8. Southern Oscillation Time Series at Tahiti and Darwin.

A novel graph based approach was recently proposed in [41, 42] where the nodes of the graph were represented by regions on the Earth and the edges were represented by the correlation between the anomaly time series of two regions. It was shown that the negative correlations are key for detecting dipoles, and thus need be preserved in both sign and magnitude. The approach discovered dipoles using a Shared Reciprocal Nearest Neighbor (SRNN) algorithm and even enabled tracking the movements of these dipoles and studying their interactions in a principled fashion. Data-guided dipole discovery techniques thus offer better predictive ability of temperature and precipitation anomalies and can be used for understanding various General Circulation Models (GCMs).

6. Concluding Remarks

Sensor datasets that are used in earth science research provide vast amounts of accurate, timely and reliable information about earth's com-

plex system. Understanding and monitoring earth's phenomena and processes require the development of novel analysis tools and techniques that are cognizant of unique data-centric issues and challenges specific to the earth science domain, such as spatial heterogeneity, multi-scale nature and uncertainty. Earth science applications using sensor datasets broadly include (i) event detection either on land, ocean or atmosphere, e.g. monitoring land cover changes using vegetation data, and identifying ocean eddy dynamics using altimeter data, (ii) relationship mining between spatio-temporal attributes and events, e.g. discovering atmospheric teleconnections such as climate dipoles. With the advancing rate of earth science sensor data acquisition technologies both at larger temporal and spatial scales, as well as the advances in computational tools and techniques, earth science research offers fertile grounds for accelerated knowledge discovery about the earth's complex system at large.

7. Acknowledgments

This work was supported in part by the National Science Foundation under Grants IIS-1029711 and IIS-0905581, an NSF Graduate Research Fellowship, an NSF Nordic Research Opportunity, and the Norwegian National Research Council as well as the Planetary Skin Institute. Access to computing facilities was provided by the University of Minnesota Supercomputing Institute.

References

[1] Atmospheric InfraRed Sounder v5 (AIRS). `http://disc.sci.gsfc.nasa.gov/AIRS/documentation/v5_docs/AIRS_V5_Release_User_Docs/V5_Data_Release_UG.pdf`.

[2] S. Boriah, V. Kumar, M. Steinbach, C. Potter, and S. Klooster. Land cover change detection: A case study. In *KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 857–865. ACM, 2008.

[3] S. Boriah, V. Kumar, M. Steinbach, P.-N. Tan, C. Potter, and S. Klooster. Detecting ecosystem disturbances and land cover change using data mining. In H. Kargupta, J. Han, P. Yu, R. Motwani, and V. Kumar, editors, *Next Generation of Data Mining*. CRC Press, 2009.

[4] S. Boriah, V. Mithal, A. Garg, V. Kumar, M. Steinbach, C. Potter, and S. Klooster. A comparative study of algorithms for land cover change. In *CIDU'10: Proceedings of Annual Conference on Intelligent Data Understanding*, pages 175–188, October 2010.

[5] S. Boriah, V. Mithal, A. Garg, M. Steinbach, V. Kumar, C. Potter, S. Klooster, and J. Castilla-Rubio. Automated detection of forest cover changes. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 44–47. IEEE, 2010.

[6] H. Cao, N. Mamoulis, and D. Cheung. Discovery of collocation episodes in spatiotemporal data. In *ICDM'06: Proceedings of the 6th IEEE International Conference on Data Mining*, pages 823–827. IEEE, 2006.

[7] Centre National d'Etudes Spatiales (CNES) and Vlaamse Instelling voor Technologisch Onderzoek (VITO). SPOT VEGETATION: Global 10 day 1 km MVC NDVI Data Set. `http://free.vgt.vito.be/`.

[8] Y. Chamber, A. Garg, V. Mithal, I. Brugere, M. Lau, V. Krishna, S. Boriah, M. Steinbach, V. Kumar, C. Potter, and S. Klooster. A novel time series based approach to detect gradual vegetation changes in forests. In *NASA Conference on Intelligent Data Understanding*, 2011.

[9] J. C. W. Chan, K. P. Chan, and A. G. O. Yeh. Detecting the nature of change in an urban environment: A comparison of machine learning algorithms. *Photogrammetric Engineering and Remote Sensing*, 67(2):213–226, Feb 2001.

[10] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection : A survey. *ACM Computing Surveys*, 41(3):15, July 2009.

[11] D. Chelton, P. Gaube, M. Schlax, J. Early, and R. Samelson. The influence of nonlinear mesoscale eddies on near-surface oceanic chlorophyll. *Science*, 334(6054):328–332, 2011.

[12] D. Chelton, M. Schlax, and R. Samelson. Global observations of nonlinear mesoscale eddies. *Progress in Oceanography*, 2011.

[13] D. Chelton, M. Schlax, R. Samelson, and R. de Szoeke. Global observations of large oceanic eddies. *Geophysical Research Letters*, 34:L15606, 2007.

[14] D. Chelton and S. Xie. Coupled ocean-atmosphere interaction at oceanic mesoscales. *Oceanography*, 23(4):52–69, 2010.

[15] J. Chen and A. Gupta. On change point detection and estimation. *Communications in Statistics: Simulation & Computation*, 30(3):665–697, 2001.

[16] K. Chen, C. Huo, Z. Zhou, H. Lu, and J. Cheng. Semi-supervised change detection via Gaussian processes. In *IGARSS'09: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages 996–996, 2009.

[17] Y. Chen, J. Randerson, D. Morton, R. DeFries, G. Collatz, P. Kasibhatla, L. Giglio, Y. Jin, and M. Marlier. Forecasting fire season severity in South America using Sea Surface temperature anomalies. *Science*, 334(6057):787–791, 2011.

[18] M. C. Chuah and F. Fu. ECG anomaly detection via time series analysis. In *WISH'07 Proceedings of the Workshop on Intelligent Systems & Smart Home*, pages 123–135, 2007.

[19] Climate Research Unit (CRU). `http://www.cru.uea.ac.uk`.

[20] P. Coppin, I. Jonckheere, K. Nackaerts, B. Muys, and E. Lambin. Digital change detection methods in ecosystem monitoring: a review. *International Journal of Remote Sensing*, 25(9):1565–1596, 2004.

[21] D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. *Proceedings of the International Conference on Intelligent Systems*, pages 82–87, 1996.

[22] E. Dwyer, J. Pereira, J. Grégoire, and C. DaCamara. Characterization of the spatio-temporal patterns of global fire activity using satellite imagery for the period April 1992 to March 1993. *Journal of Biogeography*, 27(1):57–69, 2000.

[23] J. Faghmous, Y. Chamber, S. Boriah, S. Liess, V. Kumar, F. Vikebø, and M. dos Santos Mesquita. A novel and scalable spatio-temporal technique for ocean eddy monitoring. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[24] Y. Fang, A. R. Ganguly, N. Singh, V. Vijayaraj, N. Feierabend, and D. T. Potere. Online change detection: Monitoring land cover from remotely sensed data. In *ICDM Workshops*, pages 626–631, 2006.

[25] K. Fernandes, W. Baethgen, S. Bernardes, R. DeFries, D. DeWitt, L. Goddard, W. Lavado, D. Lee, C. Padoch, M. Pinedo-Vasquez, et al. North tropical Atlantic influence on western Amazon fire season variability. *Geophysical Research Letters*, 38(12):L12701, 2011.

[26] L. Fu, D. Chelton, P. Le Traon, and R. Morrow. Eddy dynamics from satellite altimetry. *Oceanography*, 23(4):14–25, 2010.

[27] J. García-Serrano, B. Rodríguez-Fonseca, I. Bladé, P. Zurita-Gotor, and A. de La Cámara. Rotational atmospheric circulation during north Atlantic-European winter: the influence of ENSO. *Climate dynamics*, 37(9):1727–1743, 2011.

[28] A. Garg, L. Manikonda, S. Kumar, V. Krishna, S. Boriah, M. Steinbach, V. Kumar, D. Toshniwal, C. Potter, and S. Klooster. Model-free time series segmentation approach for land cover change detec-

tion. In *CIDU'11: Proceedings of the 2011 NASA Conference on Intelligent Data Understanding*, 2011.

[29] A. Garg, V. Mithal, Y. Chamber, I. Brugere, V. Chaudhari, M. Dunham, V. Krishna, S. Krishnamurthy, S. Vangala, S. Boriah, M. Steinbach, V. Kumar, A. Cho, J. Stanley, T. Abraham, J. C. Castilla-Rubio, C. Potter, and S. Klooster. GOPHER: Global observation of planetary health and ecosystem resources. In *IGARSS'11: Proceedings of the IEEE Geoscience and Remote Sensing Symposium*, 2011.

[30] German Federal Institute of Hydrology (Bundesanstalt für Gewässerkunde or BfG). Global Runoff Data Centre. `http://www.bafg.de/cln_030/nn_266918/GRDC/EN/`.

[31] L. Giglio, T. Loboda, D. Roy, B. Quayle, and C. Justice. An active-fire based burned area mapping algorithm for the MODIS sensor. *Remote Sensing of Environment*, 113(2):408–420, 2009.

[32] J. Gillis. With deaths of forests, a loss of key climate protectors. *The New York Times*, October 1 2011.

[33] Goddard Earth Observing System Data Assimilation System Version 5 (GEOS-5). `https://gmao.gsfc.nasa.gov/systems/geos5/`.

[34] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *ICDM '01: Proceedings of the IEEE International Conference on Data Mining*, pages 203–210, 2001.

[35] Y. Huang, L. Zhang, and P. Zhang. A framework for mining sequential patterns from spatio-temporal event data sets. *IEEE Transactions on Knowledge and Data Engineering*, 20(4):433–448, 2008.

[36] T. Idé and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *KDD '04: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 440–449. ACM Press, 2004.

[37] C. Inclán and G. C. Tiao. Use of cumulative sums of squares for retrospective detection of changes of variance. *Journal of the American Statistical Association*, 89(427):913–923, 1994.

[38] Japan Aerospace Exploration Agency (JAXA) and National Institute for Environmental Studies (NIES). Greenhouse gases Observing SATellite (GOSAT) Project. `http://data.gosat.nies.go.jp`.

[39] T. Kasetkasem and P. Varshney. An image change detection algorithm based on Markov random field models. *IEEE Transactions on Geoscience and Remote Sensing*, 40(8):1815–1823, 2002.

[40] Y. Kawahara, T. Yairi, and K. Machida. Change-point detection in time-series data based on subspace identification. In *ICDM 2007: Proceedings of the 7th IEEE International Conference on Data Mining*, pages 559–564, 2007.

[41] J. Kawale, M. Steinbach, and V. Kumar. Discovering dynamic dipoles in climate data. In *SIAM Conference on Data Mining, SDM*, pages 107–118, 2011.

[42] Kawale, J. and Liess, S. and Kumar, A. and Steinbach, M. and Ganguly, A. and Samatova, N.F. and Semazzi, F. and Snyder, P. and Kumar, V. Data guided discovery of dynamic climate dipoles. In *NASA Conference on Intelligent Data Understanding*, 2011.

[43] E. Keogh, J. Lin, S. Lee, and H. Herle. Finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*, 11(1):1–27, 2007.

[44] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247. ACM, 2003.

[45] J. Kucera, P. Barbosa, and P. Strobl. Cumulative sum charts - a novel technique for processing daily time series of MODIS data for burnt area mapping in Portugal. In *MultiTemp 2007: International Workshop on the Analysis of Multi-temporal Remote Sensing Images*, pages 1–6, July 2007.

[46] S. Lhermitte, J. Verbesselt, I. Jonckheere, K. Nackaerts, J. A. N. van Aardt, W. W. Verstraeten, and P. Coppin. Hierarchical image segmentation based on similarity of NDVI time series. *Remote Sensing of Environment*, 112(2):506–521, 2008.

[47] S. Lhermitte, W. W. Verstraeten, J. Verbesselt, and P. Coppin. Spatio-temporal segmentation based on subsequences of satellite image time series. *IGARSS 2007: IEEE International Geoscience and Remote Sensing Symposium*, 2007.

[48] J. Limpens, F. Berendse, C. Blodau, J. Canadell, C. Freeman, J. Holden, N. Roulet, H. Rydin, and G. Schaepman-Strub. Peatlands and the carbon cycle: from local processes to global implications – a synthesis. *Biogeosciences*, 5(2):1379–1419, 2008.

[49] Ministry of Economy, Trade, and Industry (METI) of Japan. Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Global Digial Elevation Model (GDEM). `http://www.gdem.aster.ersdac.or.jp/`.

[50] V. Mithal, S. Boriah, A. Garg, M. Steinbach, V. Kumar, C. Potter, S. Klooster, and J. C. Castilla-Rubio. Monitoring global forest cover using data mining. *ACM Transactions on Intelligent Systems and Technology*, 2(4), 2011.

[51] V. Mithal, A. Garg, I. Brugere, S. Boriah, V. Kumar, M. Steinbach, C. Potter, , and S. Klooster. Incorporating natural variation into time series-based land cover change identification. In *CIDU'11: Proceedings of the 2011 NASA Conference on Intelligent Data Understanding*, 2011.

[52] Modern-Era Retrospective analysis for Research and Applications (MERRA). `https://gmao.gsfc.nasa.gov/merra`.

[53] P. Mohan, S. Shekhar, J. Shine, and J. Rogers. Cascading spatio-temporal pattern discovery. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[54] P. Mohan, S. Shekhar, J. A. Shine, and J. P. Rogers. Cascading spatio-temporal pattern discovery: A summary of results. In *SDM'10: Proceedings of 10th SIAM International Data Mining*, pages 327–338, 2010.

[55] National Aeronautics and Space Administration. MODIS Land Data. `http://modis-land.gsfc.nasa.gov/`.

[56] National Aeronautics and Space Administration, University of Lille 1, le Centre national d'études spatiales (CNES), and L' Institut national des sciences de l'Univers (CNRS-INSU). AErosol RObotic NETwork (AERONET). `http://aeronet.gsfc.nasa.gov/`.

[57] National Climatic Data Center (NCDC). `http://www.ncdc.noaa.gov/oa/climate/ghcn-daily/`.

[58] National Oceanic and Atmospheric Administration, Global Monitoring Division. Carbon Cycle Greenhouse Gases. `http://www.esrl.noaa.gov/gmd/ccgg/`.

[59] National Oceanic and Atmospheric Administration; U.S. Geological Survey. AVHRR Normalized Difference Vegetation Index (NDVI) Composites. `http://eros.usgs.gov/#/Find_Data/Products_and_Data_Available/NDVI`.

[60] S. E. Page, F. Siegert, J. O. Rieley, H.-D. V. Boehm, A. Jaya, and S. Limin. The amount of carbon released from peat and forest fires in Indonesia during 1997. *Nature*, 420(6911):61–65, 11 2002.

[61] P. Pellikka, M. Lötjönen, M. Siljander, and L. Lens. Airborne remote sensing of spatiotemporal change (1955-2004) in indigenous and exotic forest cover in the Taita Hills, Kenya. *International Jour-*

nal of Applied Earth Observation and Geoinformation, 11(4):221–232, 2009.

[62] C. S. Potter, P.-N. Tan, M. Steinbach, S. A. Klooster, V. Kumar, R. Myneni, and V. Genovese. Major disturbance events in terrestrial ecosystems detected using global satellite data sets. *Global Change Biology*, 9(7):1005–1021, 2003.

[63] D. Preston, P. Protopapas, and C. Brodley. Event discovery in time series. In *SDM 2009: Proceedings of the 9th SIAM International Conference on Data Mining*, 2009.

[64] Reanalysis Intercomparison and Observations (RIO). `http://reanalyses.org`.

[65] M. Schroder, H. Rehrauer, K. Seidel, and M. Datcu. Spatial information retrieval from remote-sensing images. II. Gibbs-Markov random fields. *IEEE Transactions on Geoscience and Remote Sensing*, 36(5):1446–1455, 1998.

[66] M. Simard, W. Romme, J. Griffin, and M. Turner. Do mountain pine beetle outbreaks change the probability of active crown fire in lodgepole pine forests? *Ecological Monographs*, 81(1):3–24, 2011.

[67] SPARC Data Center. `http://www.sparc.sunysb.edu/`.

[68] N. Sugiura and R. T. Ogden. Testing change-points with linear trend. *Communications in Statistics - Simulation and Computation*, 23(2):287–322, 1994.

[69] W. Tober. A computer movie simulating urban growth in the Detroit region. *Economic Geography*, 46(2):234–240, 1970.

[70] University of Wisconsin-Madison, Center for Sustainability and the Global Environment (SAGE). Global river discharge database. `http://www.sage.wisc.edu/riverdata/`.

[71] U.S. Geological Survey. Landsat missions. `http://landsat.usgs.gov/`.

[72] U.S. Geological Survey. National Geospatial Intelligence Agency (NGA) / National Aeronautics and Space Administration (NASA) Shuttle Radar Topography Mission (SRTM). `http://eros.usgs.gov/#/Find_Data/Products_and_Data_Available/SRTM`.

[73] M. H. Vellekoop and J. M. C. Clark. A nonlinear filtering approach to changepoint detection problems: Direct and differential-geometric methods. *SIAM Review*, 48(2):329–356, 2006.

[74] J. Verbesselt, R. Hyndman, A. Zeileis, and D. Culvenor. Phenological change detection while accounting for abrupt and gradual trends in satellite image time series. *Remote Sensing of Environment*, 114(12):2970 – 2980, 2010.

[75] F. Verhein. k-STARs: Sequences of spatio-temporal association rules. In *ICDM Workshops*, pages 387–394, 2006.

[76] F. Verhein. Mining complex spatio-temporal sequence patterns. In *SDM'09: Proceedings of the SIAM International Conference on Data Mining*, page 605, 2009.

[77] M. Vuran, Ö. Akan, and I. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[78] World Meteorological Organization, Global Atmosphere Watch. World data centre for aerosols. `http://ebas.nilu.no/`.

[79] J. Yin, D. Hu, and Q. Yang. Spatio-temporal event detection using dynamic conditional random fields. In *IJCAI'09: Proceedings of the International Joint Conference on Artificial Intelligence*, volume 9, 2009.

[80] Y. Zhao, L. Zhang, P. Li, and B. Huang. Classification of high spatial resolution imagery using improved Gaussian Markov random-field-based texture features. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1458–1468, 2007.

Index