KDD Support Services Based on Data Semantics

Claudia Diamantini (Contact author), Domenico Potena, Maurizio Panti

Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione,
Università Politecnica delle Marche
via Brecce Bianche, 60131 Ancona, Italy
{diamanti,d.potena,panti}@diiga.univpm.it

Abstract. The identification of valid, novel and interesting models from large volumes of data is the primary goal of Knowledge Discovery in Databases (KDD). In order to successfully achieve such a complex goal, many kinds of semantic information about the KDD and business domains is necessary. In this paper, we present an approach to the characterization of semantic domain information for a particular kind of KDD process: classification. In particular we show how, by estimating the properties of the true but unknown classification model, one can derive domain information on the classification problem at hand. We discuss how, by saving these properties with the data, users profit from this information and save time for experimenting with a lot of classifiers and parameters by accessing this knowledge.

Key words: Data Mining, Data Semantics, Classification, Decision Border, User Support

1 Introduction

Knowledge Discovery in Databases (KDD) emerged as a rapidly growing interdisciplinary field that merges together databases, statistics, machine learning and related areas in order to extract valuable information and knowledge in large volumes of data. KDD aims to overcome the limitations of traditional database queries, and of the more recent OLAP techniques, in order to support analysis and decision-making. These techniques can in fact help to extract information conforming to a predefined, previously known data model, but they do not allow us to identify novel, interesting models in data. For instance, these techniques cannot help in answering the following query: "Find network connection records indicating an intrusion", just because we do not have a model of what an intrusion is. However, even if our model of connection (e.g., the set of attributes 'connection length', 'protocol', 'service', 'number of failed login attempts', 'number of root accesses' and 'number of connections to the same host') does not explicitly contain a model of intrusion, we can assume that the latter can be established from the former, in terms of typical patterns representing relations among the basic model attributes. For instance, "If ('number of connections to the same host' ≥ 10) and ('protocol' = UDP) and ('service' = echo)

and ('number of pending connections' = 'number of connections to the same host') Then (Prediction = DoS)" can be the model of one type of intrusion. This rationale underpins KDD, which studies techniques and methodologies to reveal unknown relations from available data. More formally, we define KDD as "the process of identifying valid, novel, potentially useful and ultimately understandable patterns/models in data", where data are defined as a set of facts F described by a database schema and patterns are defined as "an expression E in some language L describing facts in a subset F_E of F" [10, chap. 1]. Notice that, since patterns should be valid and potentially useful, an expression E should not limit itself to the description of the database instances, it should rather be capable of describing any new instance that could at any time be added to the database. In other words, the expression E describes the real phenomenon of which database instances are particular realizations. We often synthesize this by saying that KDD is a *model induction* activity. Different languages L define different kinds of models that can be induced. Models can be basically split into predictive ones (e.g., classification or regression models) and descriptive ones (e.g., clustering models or association rules). In the following we will consider classification models. Classification models give a description of a set of predefined classes, like e.g. the 'normal connection' and 'intrusion' classes, which are relevant for a given prediction or recognition task. Hence, in the same way a database schema defines the semantics of its instances, the schema of a classification model defines the semantics of the *classification problem*, that is of the set of classes chosen for the specific user goal (e.g., to detect intrusions).

Although a classification model is typically considered as the final result of a KDD process, it has yet another important role. As a matter of fact, to be effective, the model induction process must be guided by different kinds of domain information: information about how induction techniques work and how these can be applied in the specific business domain, the kinds of regularities one can expect to find in data and so on. In other words, we need to know the semantics of the KDD domain as well as the semantics of the business domain and how these interact. From this perspective, a classification model contains important business domain information that could be profitably used to guide the model induction process. In the next section, we will discuss the role of domain information in KDD in general, and in the classification task in particular, in deeper detail.

The contribution of this paper is an approach to characterize the semantics of classification problems, in terms of geometric properties of the classification model. The approach resorts on the relatively cheap Bayes risk weighted Vector Quantization (BVQ) learning algorithm [8] to define the analytic form of a good estimation of the true but unknown classification model. The analytical description is then used to derive some properties of the model that helps to choose probably the best classification method for the given problem, to appropriately prepare data and to train a classifier for the best classification method. By saving the analytical description of the classification model and its properties with

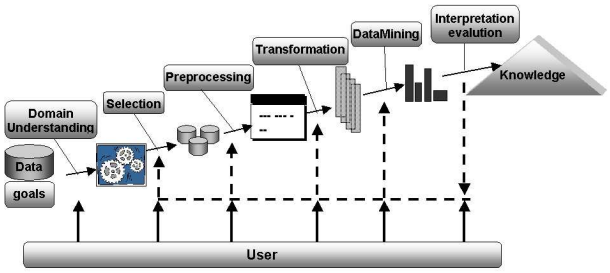


Fig. 1. The KDD process.

the data, users profit from this information and save time in experimenting with a lot of classifiers and parameters by having access to this knowledge.

The rest of the paper is organized as follows. In section 2, we briefly introduce the KDD environment, highlighting the major sources of complexity in the design of a KDD process, and introducing the main approaches proposed in the literature to overcome this complexity by exploiting information about the KDD and business domains. We then narrow the discussion on the classification problem. In section 3 we introduce the definition of nearest neighbor Vector Quantizer (VQ) and its geometrical properties, and we show how to obtain the analytical description of a VQ. Then, in section 4 we describe our VQ-based learning approach to obtain an approximation of a classification model and its analytical description. Section 5 is devoted to the discussion on the possible types of deductions on the geometry of the classification model which is derived from the analytical definition, and how the accuracy of the approximation influences the validity of the discovered knowledge. In section 6 we empirically demonstrate the validity of the approach by means of real-world classification problems. Finally, in section 7 we cast a look on the possible application of the derived information regarding the classification model in the implementation of semantic KDD support services over a net. We will come to a conclusion in Section 8.

2 The role of semantics in KDD process design

KDD as a discipline studies the definition of reference models of a process of knowledge discovery from data. According to methodological standards [10, 27], a process of discovery from data can be divided into six principal phases: Domain Understanding, Data Selection, Data Preprocessing, Transformation, Data Mining, Interpretation/Evaluation (see the schema in Figure 1). The core phase of the process is the *Data Mining* (DM) phase, where model induction is performed by one of the many available techniques. These techniques largely have a statistical foundation, and they require proper conditions of application. For this reason, the DM phase is preceded in any process model by a *data preparation* phase, that can be further split in data and attribute (feature) selection, data

preprocessing (e.g., the cleaning of data), data transformation (e.g., by normalization). Furthermore, it is known that, to be efficient and effective, statistical techniques should be guided by some model-driven hypothesis. So, the starting phase of the KDD process is devoted to domain and data understanding, where one has to find out the best representation of the business goal in terms of data mining goals, and envisage the best data structures and techniques to achieve that goal. In practice, this means drawing a rough model of the business and of the entities involved, to form some model-driven hypothesis on the kind of regularities which can be found in data and focus the search towards the most appropriate techniques. The ending phase is finally devoted to explanation and evaluation of the discovered knowledge inside the domain.

The intrinsic complexity of the design of a KDD process is due to the numerous degrees of freedom the user has to work with and to the goal-driven and domain dependent nature of the problem. When an analyst starts the discovery process, for example, she/he has to wonder: what are the database instances to select? How can I discriminate the noisy data from the informative one? Are all the data attributes equally important? Which is the best technique and algorithm to apply, and how the choice of the algorithm influences previous choices? How should one set the algorithm's parameters? Of course, the answer to these questions strictly depends on the problem and data at hand. On the other hand, the existence of a great amount of techniques and tools increases the complexity of choice, as it presupposes a certain degree of acquaintance with the mathematical theory underlying most of the techniques, so that they can be appropriately applied all together to the problem at hand, correctly and effectively used. Then, in order to design a KDD process, two kinds of expertise are needed: in the business domain and in the KDD domain. However, the user is typically a domain expert, but not a KDD expert, or viceversa. Another source of complexity is due to the intrinsic features of any discovery process, namely the lack of knowledge and consequently the difficulty to define the best plan to discover that knowledge beforehand. This fact is recognized in all the existing process models by accounting for the need of repeated backtracking to previous phases and repetition of certain actions: the knowledge acquired during a phase can suggest a revision of the choices taken at previous steps to enhance their results.

If no support is given to the user, then a blind search over the joint space of possible tasks, techniques, algorithms, parameters produces inevitably unsatisfactory results with great effort. Basically, the kind of support that a user can be given involves the following facilities:

- to understand the business domain and goal and to relate it to a suitable set of KDD tasks;
- to choose the more suitable tools for the user goals and business domain, on the basis of a number of characteristics:
  - performance (complexity, scalability, accuracy),
  - the kind of data they can be used for (textual/symbolic data, numerical data, structured data, sequences, ...),
  - the kind of goal they are written for (data cleaning, data transformation, data mining, visualization, ...),

- the kind of data mining task (classification, rule induction, ...);

  this involves facilities to browse the tool repository and to obtain information about the tools;
- to set algorithm parameters, especially for Data Mining algorithms, in the appropriate manner with respect to the problem at hand;
- to manage all kinds of data involved in the KDD process, namely, raw and structured data, intermediate data and models, in terms of access, selection, preparation of data conforming to the tool input format, and so on;
- to design the KDD process by tool composition;

Most of the previous features rely on different forms of semantic information about data, tools and business domains. In the literature, the use of domain ontologies is largely proposed to guide the KDD process and to give support to domain experts. In [18, chap.23] and [25] a business domain ontology supports the extraction of novel features, by exploiting relations among domain concepts. In [16, 32] the use of ontologies is proposed to refine the induced knowledge and to correctly interpret the results. [5] discussed the use of ontologies in the whole KDD process for the medical domain. Finally, in distributed environments, it has to be pointed out the role of business domain ontologies both in the search for appropriate data and in their integration [18, chap.23], [31]. A special kind of domain ontology is the KDD ontology. A KDD ontology is a conceptualization of the KDD domain in terms of tasks, techniques, algorithms, tools and tool properties like performance and the kind of data that can be used for [3, 20, 33]. As such, a KDD ontology has a similar role with respect to the business domain ontology: it helps the business expert to understand the KDD domain, so that he can either effectively collaborate with a KDD expert in the design of a KDD project, or design the KDD project on his own. In this case, it can support the user in browsing a tool repository organized with respect to the KDD ontology. Semantic description of tools is also adopted to support standardization and process design by tool composition [11, 14]. However, to support the proper choice and use of tools, the semantics of the problem is also needed. For instance, in the classification task, it is known that different classification techniques work better on certain classes of classification problems than others. Hence, the classification model can be exploited as a fundamental domain information to semantically guide the KDD process design. This principle is the basis of *Meta-learning*, that refers to a bulk of techniques to discover domain semantics hidden in data, which is then used to guide the choice of Data Mining algorithms [2, 17] or to form a prototype domain model that guides further investigations [30]. In the following subsection we examine this topic thoroughly, for the specific case of the classification task, by giving a more formal definition of the nature of a classification problem and discussing how it can support the user in dealing with each phase of a KDD classification process. At the end of the paper, we will also discuss the advantages of equipping data published over a net with this kind of domain semantics, in order to build case-based support services.

2.1 Evidence-Based Classification Process Design

Let us start with a formal definition of the classification problem:

Definition 1. *Given a set $O_s = \{o_1, \ldots, o_m\}$ of observation n-tuples and a set of classes $C = \{c_1, \ldots, c_k\}$, the classification problem is to define a* classification rule*, that is a mapping $\Phi : O_s \rightarrow \mathcal{C}$, where each n-tuple is assigned to a class. A class c_j contains precisely those tuples mapped to it; that is $c_j = \{o_i | \Phi(o_i) = c_j, 1 \leq i \leq m, o_i \in O_s\}$.*

Note that classes are predefined, non overlapping and they partition the entire set of n-tuples. In this sense, the classes of a classification problem are indeed *equivalence classes*.

In order to model and characterize the properties of a classification problem, we can take a geometric point of view. In a geometric model the observation n-tuples are represented as points in a $\mathcal{R}^n$ vector space. In this way, the mapping Φ defines a partition of the vector space, where each partition region defines an equivalence class. These regions are called *decision regions*, while the border between decision regions is called the *decision border* (see Figure 2).

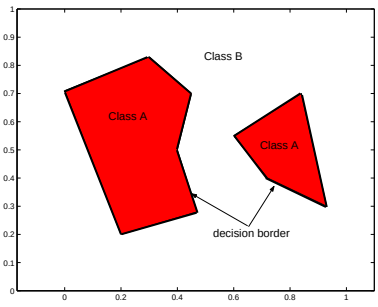


Fig. 2. A two dimensional vector space, with decision regions and the decision border for a two-class problem.

The geometric properties of the decision regions have their natural interpretation as properties of the true classes or, in other terms, classification problems can be characterized in terms of the properties of the decision regions. For instance, we speak of linearly separable classification problem, when it is possible to separate classes without error by a hyperplane. On the other hand, when the decision border is a generic (possibly non connected) curve, then we speak of non linearly separable classification problems.

Let us illustrate how the decision regions (or, equivalently, the decision border) define an evidence that can guide the choices during the whole classification process. To start with, it is straightforward to observe that a general (and generic) knowledge of the shape of the decision regions and their localization in the vector space can be very useful in the preliminary data analysis phase. As

a matter of fact, one of the most useful techniques adopted during this phase is visualization, since, it is said that, the most powerful data miner tool is the human eye. Unfortunately, visualization techniques can be directly applied only in problems of low dimensionality, say from one to three dimensions. The analytical form of the decision border could give important geometric information about the problem that cannot be visually inspected.

A particular form of data selection is called *data reduction* in the literature. It is a technique exploited in combination with non scalable algorithms, that is algorithms whose high cost make them unsuitable for the management of large amounts of data. Data reduction in classification problems can be performed by eliminating those samples falling far from the decision border, as less informative samples. This intuition dates back to the earliest work in Pattern Recognition, forming the basis of the condensed nearest neighbor method [15].

The decision border can support feature selection: in [23], a feature selection algorithm is shown which is based on the evidence that, following a direction not parallel to the decision border, the classification changes. Then, the direction normal to the decision border is the most informative one. By considering the principal components of the decision border, one can decide suitable transformations of data and the elimination of the less informative features. Unfortunately, the paper illustrates a method to derive the principal components that is computationally heavy and does not scale well.

The selection of the classifier architecture and the learning algorithm is based on the geometry of the decision border in the data mining step. For instance it is known that, if the border turns out to be parallel to the axes, then one can decide to use decision trees, that perform well on these problems and that have the advantage of a simple rule extraction. Other kinds of linear borders can suggest the use of Support Vector Machines (SVM) with linear kernel. Similarly, closed and convex decision regions would turn the choice towards SVM with gaussian kernel or Radial Basis Function (RBF) networks, while for open, non linear decision borders SVM with polynomial kernel or Multi-Layer Perceptron (MLP) would be preferable. To set the number of layers in MLP it is useful to know the type of concavity of the decision regions as well as the number of disconnected regions. One could even envisage a combined method where different types of architectures are used in different regions of the vector space, depending on the form of the border in that region. Finally, the initial state of a learning algorithm can be set in the regions of space near the decision border.

In the elicitation of the classification rule, the analytical representation of the decision border finds its natural application. It is known that one of the perceived limits of inductive techniques such as neural networks is their "black-box" nature: the classification rule is hidden in the structure of the network and a human expert has no element to validate it. In the literature, different techniques have been proposed to extract rules from MLP neural networks and decision trees. The method described in section 4 defines the basis for the elicitation of VQ-based classification rules.

3 Generalities on Nearest Neighbor Vector Quantizers

Definition 2. *A nearest neighbor Vector Quantizer (VQ) of dimension n and order M is a function* $\Omega : \mathcal{R}^n \rightarrow \mathcal{M},\ \mathcal{M} = \{m_1, m_2, \ldots, m_M\}, m_i \in \mathcal{R}^n, m_i \neq m_j$*, which defines a partition of* $\mathcal{R}^n$ *into M regions* $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_M$*, such that*

$$\mathcal{V}_i = \{x \in \mathcal{R}^n : d(x, m_i) < d(x, m_j),\ j \neq i\}, \tag{1}$$

where d is some distance measure.

$\mathcal{M}$ is called the *code*. It is a finite set of vectors in $\mathcal{R}^n$ called *code vectors* or *reference vectors*. The region $\mathcal{V}_i$ defined by (1) is called the *Voronoi region* of the code vector m_i.
Notice that, once the distance has been defined, $\mathcal{M}$ defines entirely the mapping Ω. If we choose as distance measure the usual squared Euclidean distance

$$d(x, y) = \parallel x - y \parallel^2 = (x - y)^T (x - y),\ x, y \in \mathcal{R}^n, \tag{2}$$

then it is particularly simple to describe the partition as a function of code vectors. In practice, we can reduce the definition of the Voronoi region $\mathcal{V}_i$ to the following system of constraints:

$$\mathcal{V}_i : \begin{cases} \parallel x - m_i \parallel^2 < \parallel x - m_{ci_1} \parallel^2 \\ \qquad\qquad \vdots \\ \parallel x - m_i \parallel^2 < \parallel x - m_{ci_l} \parallel^2, \end{cases} \tag{3}$$

where $Neigh(m_i) = \{m_{ci_1}, \ldots, m_{ci_l}\} \subset \mathcal{M}$ is the set of nearest code vectors to m_i. Region borders are defined as the geometric locus of points equidistant from at least a pair of code vectors. In particular, the region border is a piecewise linear surface, where each piece of hyperplane (excluding the extreme points) satisfies with the equal sign exactly one of the constraints in (3). Thus, if l is the number of constraints, $\mathcal{V}_i$ is a polytope with l faces. Figure 3 gives an illustrative example of a code of order 10 and of the relative Voronoi diagram in $\mathcal{R}^2$. Points

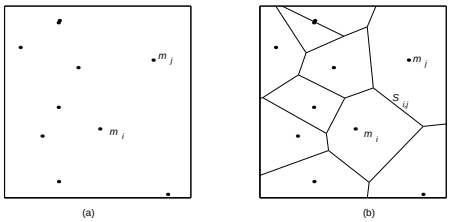


Fig. 3. (a) A code and (b) its Voronoi diagram.$\mathcal{S}_i, j$ is the border between m_i and m_j.

satisfying two or more constraints with the equal sign define the vertices of the

polytope $\mathcal{V}_i$. Finally, let us notice that each constraint in (3) defines a half-space of the type

$$V_{i,j} = \{x \in \mathcal{R}^n : u_{ij}^T \cdot (x - \beta_{ij}) \geq 0\},$$

with $u_{ij} = m_i - m_j$ and $\beta_{ij} = \frac{m_j + m_i}{2}$. Each region $\mathcal{V}_i$ is thus defined by the intersection of a finite number of half-spaces, hence it is a regular, convex polytope. It is also simple to see that each code vector m_i belongs to the region $\mathcal{V}_i$. In fact, it holds $\| m_i - m_i \|^2 < \| m_i - m_j \|^2$ for each code vector $m_j \neq m_i$, then $m_i \in \mathcal{V}_i$. Regular VQs of this type are called polytopal VQs.

3.1 Voronoi Diagrams in n-dimensional space

In spite of the simple and well known theory illustrated above, few or no software for the practical calculus of Voronoi diagrams *in spaces of general dimensionality* $\mathcal{R}^n$, $n > 2$ exists (actually, the only one we found, which only approaches our needs is reported in [7]).

We implemented such an algorithm, that is available as a web service at *http://babbage.diiga.univpm.it:8080/axis/services/voronoiWrapped*. The WSDL of the service can be downloaded at *http://babbage.diiga.univpm.it:8080/axis/-WSDL/voronoiWSDL.xml*. Definitions and technical details are given in Appendix A.

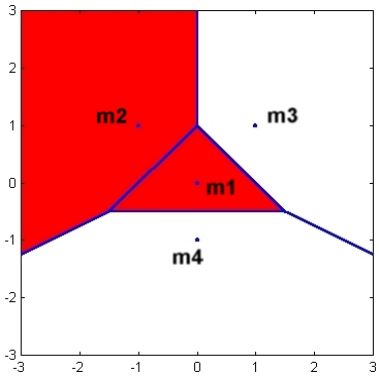
In Figure 4 a graphical example of a Voronoi diagram in $\mathcal{R}^2$ and the corresponding output of the algorithm is given . For each code vector m_i the equations of the pieces of lines and their extremes is reported (the points in the $from \ldots to \ldots$ expression). $X(i)$ represents the i-th dimension of the feature space. Notice the existence of extremes with very big values not comparable to values of the code vectors. These represent the approximation of "points at infinite" that are introduced for computational purposes and which are called *fictitious code vectors* in the Appendix. Notice that the use of fictitious code vectors introduces approximation errors in some equation (e.g., the 3rd equation of m2 should be X(1)-2*X(2)=-0.5).

4 An Approach to Decision Border Characterization

Statistical pattern classification is modeled by considering a pair $(\mathbf{x}, \mathbf{c})$ of random variables with values in $\mathcal{R}^n \times \mathcal{C}$. The continuous vector $\mathbf{x}$ is the *observed* vector (or *feature* vector), while the discrete random variable $\mathbf{c} \in \mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ is the *class* the observed vector belongs to. Each class c_i is characterized by a conditional density function $p_{\mathbf{x}|\mathbf{c}}(\mathbf{x} = x|\mathbf{c} = c_i)$, and by an *apriori probability* $P_{\mathbf{c}}(c_i)$, $\sum_{i=1}^{k} P_{\mathbf{c}}(c_i) = 1$. The best theoretical rule to assign a feature vector to a class is known as the *Bayes rule*. It reads:

$$c^* = argmax_{c_i}\{P_{\mathbf{c}}(c_i) * p_{\mathbf{x}|\mathbf{c}}(\mathbf{x} = x|\mathbf{c} = c_i)\}.$$

This rule produces in fact the minimum misclassification rate. For this reason, decision borders defined by the Bayes rule are considered the *true* (Bayes) decision border for the problem.



The Region of code vector m1 (0.000,0.000) label 2:
1.000*X(1) + -1.000*X(2) = -1.000 from (-0.000,1.000) to (-1.500,-0.500)
1.000*X(1) + 1.000*X(2) = 1.000 from (-0.000,1.000) to (1.500,-0.500)
0.000*X(1) + 1.000*X(2) = -0.500 from (-1.500,-0.500) to (1.500,-0.500)
The Region of code vector m2 (-1.000,1.000) label 2:
1.000*X(1) + -1.000*X(2) = -1.000 from (-0.000,1.000) to (-1.500,-0.500)
1.000*X(1) + 0.000*X(2) = -0.000 from (-0.000,1.000) to (-0.000,1001.000)
1.000*X(1) + -2.001*X(2) = -0.499 from (-1.500,-0.500) to (-667.059,-333.274)
The Region of code vector m3 (1.000,1.000) label 1:
1.000*X(1) + 1.000*X(2) = 1.000 from (-0.000,1.000) to (1.500,-0.500)
1.000*X(1) + 0.000*X(2) = -0.000 from (-0.000,1.000) to (-0.000,1001.000)
1.000*X(1) + 2.001*X(2) = 0.499 from (1.500,-0.500) to (667.059,-333.274)
The Region of code vector m4 (0.000,-1.000) label 1:
0.000*X(1) + 1.000*X(2) = -0.500 from (-1.500,-0.500) to (1.500,-0.500)
1.000*X(1) + -2.001*X(2) = -0.499 from (-1.500,-0.500) to (-667.059,-333.274)
1.000*X(1) + 2.001*X(2) = 0.499 from (1.500,-0.500) to (667.059,-333.274)

Fig. 4. An example of analytical description of a Voronoi Diagram and its graphical representation.

The form of the true decision border is generally unknown, since the Bayes rule is based on the definition of the unknown class conditional distributions. In the following we describe a method to estimate the form of the true but unknown decision border. It relies on the Labeled Vector Quantizer (LVQ) architecture as a classification architecture and on the BVQ algorithm [8] to design an LVQ approaching the Bayes rule.

Definition 3. *A Labeled Vector Quantizer (LVQ) is a pair* $LVQ =< \Omega, \mathcal{L} >$, *where* $\Omega : \mathcal{R}^n \rightarrow \mathcal{M}$ *is a vector quantizer, and* $\mathcal{L} : \mathcal{M} \rightarrow \mathcal{C}$ *is a labeling function, assigning to each code vector in* $\mathcal{M}$ *a class label.*

An LVQ defines a decision rule:

Definition 4. *The decision rule associated with a Labeled Vector Quantizer* $LVQ =< \Omega, \mathcal{L} >$ *is:*

$$\Phi_{LVQ} : \mathcal{R}^n \rightarrow \mathcal{C}, x \mapsto \mathcal{L}(\Omega(x)).$$

Notice the nearest neighbor nature of this decision rule: each vector in $\mathcal{R}^n$ is assigned to the same class as its nearest code vector. Thus, decision regions

are defined by the union of Voronoi regions of code vectors with the same label. Notice also that decision borders are defined only by those hyperplanes $\mathcal{S}_{i,j}$ such that m_i and m_j have different labels.

The design of an LVQ decision rule approaching the Bayes rule is practically realized by supervised inductive learning algorithms, based on a set of examples of known class. Among the learning algorithms for LVQ classifier design, BVQ turns out to be the one with the best overall performances [8].

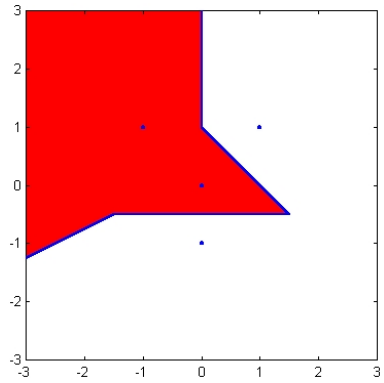
In the following we summarize the steps of the proposed method to extract the analytical description of the decision border. The method is valid, and we tested it, in general $\mathcal{R}^n$ space and for k-class problems. However in the following, for sake of simplicity and to give a visual support to the reader, the examples are given considering two-class problems in a two-dimensional space.

Let $T = \{(x_1, l_1), \ldots, (x_N, l_N)\}$ be a set of N labeled samples, where x_i is the feature vector and $l_i \in \{c_1, \ldots, c_k\}$ is its class.

1. **BVQ Training**: Use the N samples to train an LVQ. In this phase we have to set up the parameters of the LVQ and BVQ algorithm, that are principally the number of code vectors, the width of the windows Δ, the learning rate γ and the number of iterations. The tuning of the BVQ parameters is usually done experimentally, by trying different n-tuples of parameter values and evaluating the classification error for each. The total time and effort we put into the parameter setting depends on the quality of the classifier that we want to design and, of course, on the difficulty of the classification problem. The output of this step is the set of code vectors used to approach the Bayes border.
2. **Analytical Voronoi Description**: Apply the algorithm described in Section 3.1 and Appendix A to the trained code vectors, to obtain the equations of hyperplanes and circumcenters representing the Voronoi border surfaces and their vertices respectively.
3. **Decision border extraction**: Starting from the equations of the Voronoi diagram, obtain the decision border description by deleting all the borders dividing two regions with the same label. This can be done in practice by merging the pieces of hyperplane of all the code vectors with the same label, deleting those appearing twice. The result of this step for the example in Figure 4 is reported in Figure 5.

5 Analysis

Having the analytical definition of decision borders, one can develop any kind of geometrical analysis. We hasten to point out that the results of the analysis depend on the quality of the classifier w.r.t. (1) correctness and (2) simplicity. Correctness is related to the accuracy in Bayes decision border approximation. We will show that valid analysis can be carried out, even if the classifier is not accurately designed. More refined analysis cannot be guaranteed to be valid unless the classifier is near-optimal. Simplicity of the classifier depends on the



Decision Border:
1.000*X(1) + 0.000*X(2) = 0.000 from (-0.000,1.000) to (-0.000,1001.000)
1.000*X(1) + 1.000*X(2) = 1.000 from (-0.000,1.000) to (1.500,-0.500)
0.000*X(1) + 1.000*X(2) = -0.500 from (-1.500,-0.500) to (1.500,-0.500)
1.000*X(1) + -2.001*X(2) = -0.499 from (-1.500,-0.500) to (-667.059,-333.274)

Fig. 5. An example of Decision Border equations.

number of constraints used to describe the decision border, and hence indirectly on the number of code vectors. Of course, following the Occam's razor principle, such number should not be greater than the minimum number of constraints necessary to guarantee a given level of correctness, since each constraint beyond this number contributes reducing the human understanding of the model. In the following we will consider different types of analysis drawing attention to these issues.

5.1 Topological Properties of Regions

One simple analysis is that on the qualitative shape of the curve, that is of topological properties such as the Connected/Disconnected and the Open/Close properties of the decision regions. This information is derivable simply from the analysis of vertices. A decision region is Connected (Disconnected) if, denoted by V any of its vertices, it is (not) possible, starting from V, to pass along all the other vertices, moving along the pieces of hyperplanes of the region surface. To evaluate the connected property of a decision region we can use any algorithm to evaluate the connection of a graph. To evaluate the open/close property of the region (or of its sub-regions, if it is disconnected), it is sufficient to evaluate if a fictitious vertex exists in the set of its vertices. Since this vertex does not really exist, the region turns out to be unlimited (for instance, Figure 10 shows an example of two closed decision regions, while Figure 7 shows an example of open decision region). We illustrate the method for extracting the open and connected properties by the following simple algorithm:

let S be the list of all vertices V_i, $i \in \{1, \ldots, M\}$ of the decision regions. With a little abuse of notation, $S[i]$ will denote the i-th element of the list S,

while $|S|$ will denote the number of elements in S. Let A be the matrix of links between any pair of vertices, such that $A[i,j] = 1$ iff $S[i]$ and $S[j]$ both belong to a piece of hyperplane delimiting the decision region and $i \neq j$. Finally, let C be an array such that $C[t]$ contains the list of vertices of the region analysed at the t-th iteration.

1. $t = 0$;
2. $t = t + 1$;
3. Set $X = S[1]$ and set $l[t] =$'close';
4. $i = 0$;
5. While (($i < |S|$) and ($l[t] =$'close')) do
   (a) $i = i + 1$;
   (b) if $S[i]$ is a fictitious vertex then $X = S[i]$ and $l[t] =$'open';
6. if ($X \notin C[t]$) then insert X in $C[t]$;
7. $j = 1$;
8. While ($j \leq |S|$) do
   (a) if $A[i,j] = 1$ then do
      i. $A[i,j] = 0$ and $A[j,i] = 0$;
      ii. $X = S[j]$;
      iii. $i = j$ and $j = 1$;
      iv. if ($X \notin C[t]$) then insert X in $C[t]$;
   (b) else $j = j + 1$;
9. delete from S vertices in $C[t]$;
10. If S is not empty go to step 2;

If $t = 1$ the decision region is connected, otherwise it is disconnected and formed by t sub-regions. The pairs ($C[i]$, $l[i]$), with $i = 1, \ldots, t$ individuate the t sub-regions and if they are closed or open. For the example shown in Figure 5 the input to the algorithm can be

$$\begin{array}{l} S[1] = (-0.000, 1.000), \\ S[2] = (-0.000, 1001.000), \\ S[3] = (1.500, -0.500), \\ S[4] = (-1.500, -0.500), \\ S[5] = (-667.059, -333, 274) \end{array} , \; A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

while C is empty. At the end, the algorithm produces the following structure:

$$\begin{pmatrix} & (-0.000, 1001.000) & \\ & (-0.000, 1.000) & \\ C[1] = & (1.500, -0.500) & , \; l[1] = `open' \\ & (-1.500, -0.500) & \\ & (-667.059, -333, 274) & \end{pmatrix},$$

so we get an open, connected decision region (the grey region in Figure 5). The other decision region (the white one) is obviously obtained by complement.

For this analysis even rough classifiers allow us to obtain correct deductions. Figures 6-10 show five classification problems characterized each by a different topology of the true decision regions. In the figures the thin lines represent the true Bayes decision borders, while the thick lines are the borders found without stressing the design of the classifier by the BVQ, that is, by choosing a uselessly high number of code vectors, by running the BVQ for a limited number of iterations and without tuning the parameters γ and Δ. We can notice that the decision borders found by BVQ differ considerably from the true ones, nevertheless, topological properties are preserved. This is especially evident in Figures 7, 8 and 10.

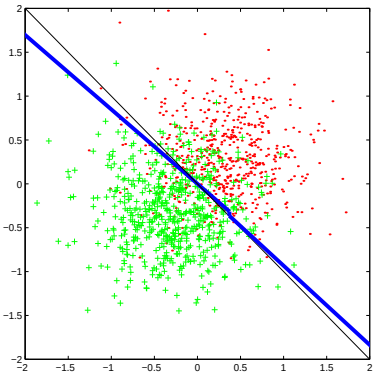


Fig. 6. A problem with a linear decision border.

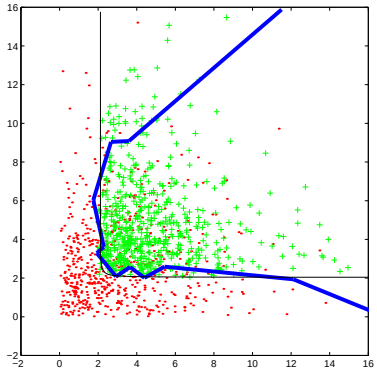


Fig. 7. A problem with a hyperbolic decision border.

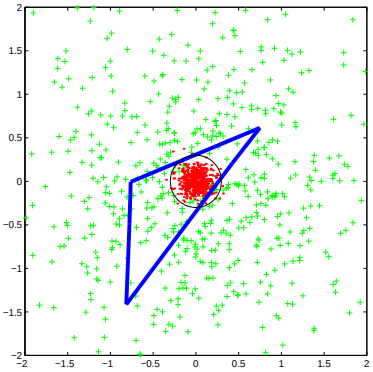


Fig. 8. A problem with a circular decision border.

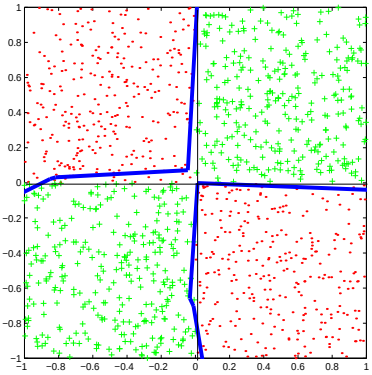


Fig. 9. The XOR problem.

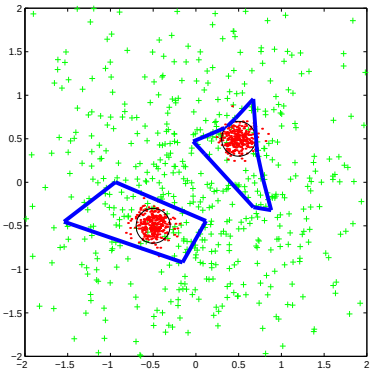


Fig. 10. A problem with a decision border formed by two disconnected circles.

5.2 Extraction of Geometrical Properties of Regions

For any (sub)regions, and specially for closed ones, we can extract a number of geometrical features to refine the regions characterization. These features include the surface area, volume, principal components ratio, convexity, volume/surface ratio, position in $\mathcal{R}^n$ and so on. Principal component analysis, in the classification domain, aims to measure the most informative direction for the classification task. Following [23], it is quite simple to calculate principal components for the piecewise linear borders of an LVQ. Convexity of decision regions can be established if, for each pair of vertices a, b their convex combination $\alpha a + (1 - \alpha)b$ belongs to the decision region, or to the decision border, for each $0 \leq \alpha \leq 1$. For lack of space, we omit the description of the algorithms to calculate all these geometrical features.

The precision of most of the geometrical features depends on the quality of the classifier. Consider for example the convexity property: from Figures 7 and 10, we can deduce that convexity is not easily preserved. Notice however that in Figures 7 and 10 convexity is not preserved only in limited and local regions of space, so that in a global analysis they can be ignored. This is true in general: further elaborations and approximations of a rough classification would allow us to enhance the quality of the deductions. Notice also that comparison between regions properties, like the relative dimensions and the relative positions of the (sub)regions remains valid (see Figure 8, 9 and 10).

6 Case Studies

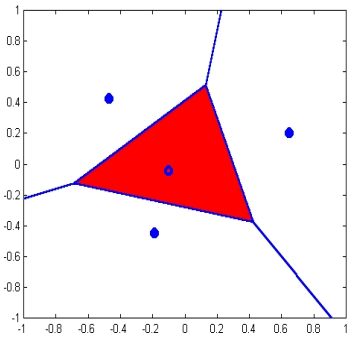
In this section we show how the knowledge given by the decision border can be exploited to support the user in the design of classification KDD processes. To this end, we consider two classification problems, the gaussian signals and the echocardiogram from the Irvine "UCI Machine Learning Repository" [1].

The former problem is to classify between two bivariate gaussian signals having the same mean and different covariance matrices. This model describes the non coherent reception of two equiprobable, noisy, binary modulated signals, and in the past years it was widely adopted as a benchmark for many recognition tasks, like feature selection [24] and classification [19]. We set the following values for means and covariance matrices:

$$\mu_1 = \mu_2 = \binom{0}{0},\ \Sigma_1 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Sigma_2 = I * 0.01$$

The Bayes border is a circle centered in the origin of axes with radius around 0.2, Bayes error probability is 2.7%. From this population, a training set TS of size 100.000 is used to design the classifiers: 50.000 to generate the model and the leftover instances to test it. First, we apply the proposed approach to extract a rough analytical description of the decision border. The outcomes are showed in Figure 11, where the BVQ is trained with 4 code vectors.

The database of the latter classification problem, extracted from the UCI repository, represents patients who suffered heart attacks at some point in the



Decision Border:
1.000*X(1) + 0.334*X(2) = 0.302 from (0.42767,-0.37584) to (0.13028,0.51438)
1.000*X(1) + 4.457*X(2) = -1.247 from (0.42767,-0.37584) to (-0.6852,-0.12613)
1.000*X(1) + -1.273*X(2) = -0.525 from (0.13028,0.51438) to (-0.6852,-0.12613)

Fig. 11. The Decision Border equations related to the two bivariate gaussian signals problem.

past. The goal of this problem is to predict whether or not the patient will survive. The most difficult part of this problem is related to the size of the data set. In fact, after the data cleaning phase, the database consists of 9 features plus the class, and it contains only 132 instances. In this case, the analytical form of the decision border is obtained initializing the BVQ with only 2 code vectors. The results are shown in Figure 12.

Decision Border:
-0.981*X(1)+ 0.130*X(2)+ -0.045*X(3)+ -0.092*X(4)+ 0.038*X(5)+
-0.004*X(6)+ -0.060*X(7)+ 0.021*X(8)+ -0.064*X(9) = -0.174

Fig. 12. The Decision Border equations related to echocardiogram database.

In the following, we show how the knowledge of the decision border helps the user in the choice of the appropriate classification algorithm, in the data selection phase and for feature selection.

6.1 Choosing the Data Mining Algorithm

Support Vector Machines is a kernel-based learning methodology introduced by Vapnik [29] that finds many important applications in the Data Mining field. Varying the form of the kernel parameter, the algorithm can simulate different classifiers, from linear classifiers, to RBF and MLP neural networks. If no knowledge about the data is given, the different kernels have to be evaluated extensively, while some knowledge about the border could limit the search to

training instances	50000	5000	2000	1000	500	100
runtime in seconds	87.44	3.38	0.65	0.22	0.07	0.03
number of support vectors	4335	1812	790	520	279	135
Accuracy on test set	97.14%	97.02%	96.95%	95.48%	94.02%	02.54%

Table 1. Using the decision border characterization for data selection.

the most promising ones. To prove this statement, let us consider the bivariate gaussian signals problem. Analyzing the equations of the decision border we see that it is a connected closed one (see Figure 11), so we deduce that it is not convenient to use a linear classifier, but we have to train a classifier by an algorithm that builds a non-linear decision rule. As a matter of fact, training the SVM algorithm with RBF kernel, we obtained an accuracy of 97.14% on the test set, and the training was carried out in 87.44 seconds. On the other hand, using a linear kernel, the same implementation of the algorithm runs on the same machine in 13683.82 seconds with an accuracy on the test set of 49.90%.

6.2 Data Selection

Data reduction techniques are exploited in combination with not scalable algorithms, that is algorithms whose high cost makes them unsuitable for the management of large amounts of data. The major information for the classification task is concentrated in the instances close to the decision border. Thus, data reduction can be performed by eliminating those samples falling far from the decision border, as less informative samples. This intuition dates back to the earliest work in Pattern Recognition, forming the basis of the condensed nearest neighbor method [15]. In the following, we show that SVM could also gain from the application of this data reduction technique. To this end, we consider again the bivariate gaussian signals problem, and we build six different experiments by training a SVM with a RBF kernel on the whole training set and on the first 100, 500, 1.000, 2.000 and 5.000 samples falling close to the decision border found by the BVQ algorithm. The results are shown in Table 1. It is noted that reducing the number of training instances from 50.000 to 2.000, the accuracy on the test set does not substantially change (it varies from 97.14% to 96.95%). On the other hand, the number of support vectors is reduced from 4.335 to 790, considerably reducing the model complexity.

Reduction in model complexity has a great impact on the training time, as shown in the Table, and also in the time needed to classify a new sample. Furthermore, the simpler the model is, the simpler it is to try to validate it and to extract symbolic rules from it.

6.3 Feature Selection

The echocardiogram database allows us to show how the characterization of the decision border provides information about the most informative features.

#	Feature	Weight (%)	Acc. W.
1	survival	0.684	0.684
2	age-at-heart-attack	0.090	0.775
4	fractional-shortening	0.064	0.839
9	mult	0.045	0.884
7	wall-motion-score	0.042	0.925
3	pericardial-effusion	0.031	0.956
5	epss	0.026	0.983
8	wall-motion-index	0.015	0.997
6	lvdd	0.003	1.000

Table 2. Weight of the features, of echocardiogram database, with respect to the classification task. Acc. W. is accumulation of weights.

Starting from the equations of the decision border (see Figure 12), we are able to extract the weight of any feature with respect to its contribution to classification accuracy. According to the EDBFE method [23], these weights can be obtained simply, by analyzing the vector normal to the decision border, that represents the most informative direction. Table 2 reports the feature information weights in decreasing order, together with the cumulative weights for all the features. Notice that, even with the simple linear border found by the BVQ, it is possible to derive results that are consistent with the domain knowledge: the most important feature to predict whether or not the patient will survive is the number of months that the patient survived immediately after the attack (the first period is clearly the most critical), while the second one is the age at which the heart attack occurred. This can be observed also empirically, since experiments performed show that training a classifier on the whole vector space and on the space formed by the 'survival' and 'age-at-heart-attack' features only, leads to the same classification accuracy.

Note that the EDBFE method extracts the information analyzing the whole data set, thus requiring a lot of computational resources, while extracting this information from the analytical form of the decision border is a straightforward and cheap operation.

The knowledge derived from this analysis can be useful for itself, giving information as to which variables mainly influence the problem, or it can be exploited to train a classifier on a limited set of features, thus reducing the so called *curse of dimensionality*.

7 Semantic Annotation of Data on a net

Business and scientific organizations can have numerous advantages in the definition of distributed KDD processes over a network: they can share data, algorithms and computational resources, as well as methodological practices. Also isolated users can exploit the network environment to retrieve and reuse useful algorithms, tools, data and discovered models.

In this perspective, a number of proposals have been made, to define effective infrastructures for KDD process design over a net. Data and Knowledge Grids have been defined as a means to support high-performance distributed data mining in federated environments [4, 6]. The service oriented paradigm is the natural extension to open environments [28, 13, 9, 22, 26, 21]. This calls for languages and standards to describe resources, in order to facilitate their discovery, comprehension, exploitation, interoperability. For instance, Grossman introduced the Predictive Model Markup Language (PMML) [14] that, in its latest version, supports the description of a classification model, as well as of data transformation activities that precedes model induction in KDD. For a survey on the development of data mining related standards see [12].

The annotation of data with semantic information about the decision border can leverage the development of a class of services of particular interest in the field of KDD, that of high-level services to give support to the users in the mapping between his business goal and the Data Mining tasks, in the choice, retrieval and correct use of techniques and tools, in their efficient composition and in the understanding of the final results. In fact, the sharing of such data, together with adopted techniques and experimental results allow us to accumulate knowledge to build the knowledge-base of an intelligent support system. In the envisaged scenario, such knowledge-base manages the relationships among three different

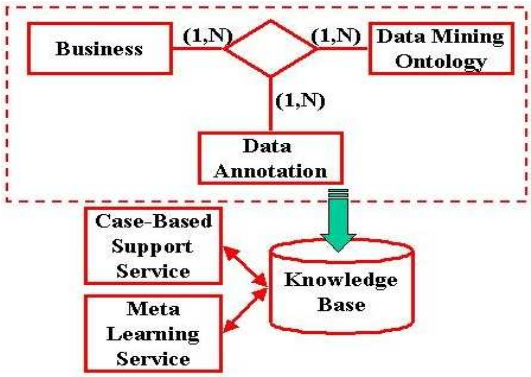


Fig. 13. The knowledge-base registry and the intelligent services of an KDD support system.

registries, representing information about the business, the data mining task and the data (Figure 13). In particular, for any performed classification experiment the knowledge-base registry stores information about the business domain, the business goal, the dataminer and his annotations, the data structure and the feature semantics, the properties of the decision border and the relations with both the methods, the algorithms and the tools used for the classification task, in accordance with a data mining ontology. The information on the decision bor-

der represents its analytical form, its geometrical and topological properties and the accuracy of the BVQ classifier used for the decision border definition. The accuracy of describing the decision border can be measured by the classification error, so if it turns out to be close to the Bayes classification error, then we can be quite confident that the decision border found is close to the Bayes border. Furthermore, the knowledge-base registry contains information about the quality of the experiment, in terms of performances of the classification algorithm: error probability, precision, recall, dimension of the inducted model, computation time and used memory. The accuracy of proposed method, the dataminer reliability and the performances of the algorithm are a measurement of the quality of the data semantic annotations. This information can be available over a network of virtual organizations as results of a single classification process or as similar distributed knowledge-base registries. Then, *Meta Learning Service* can collect and analyze this information to find similarities between data, by clustering the input datasets on the basis of their decision border characteristics. By mapping a cluster to the classification methods and algorithms used for the datasets belonging to the cluster, the meta-learning service can establish a relationship between data characteristics and algorithms performances. The results of the analysis performed by this service is also stored in the knowledge-base repository. In turn, this information can be exploited by another class of intelligent services, that of *Case-Based Support Services* which, querying the knowledge-base repository for Meta Learning information, can return the set of algorithms, or the typical parameter setting for a given algorithm, that have demonstrated the best performances on the data cluster similar in characteristics to a given dataset.

8 Conclusions

This work investigated the utility to exploit data semantics in the development of KDD processes. We considered a special kind of semantics for classification problems, given by the decision border. We showed that it is possible to derive some knowledge about the characteristics of the decision border and decision regions of a classification problem, starting from the geometrical properties of Vector Quantizers. Then, we showed that this knowledge can effectively help the user to take decisions and to limit the efforts in the implementation of inductive classification methodologies. We also discussed the introduction of intelligent services to collect, to analyze and to manage the semantics of data distributed over a network of virtual organizations.

An important issue related with this approach is the accuracy of the decision border needed for the different applications. By the term knowledge, it is commonly meant "a set of assertions about a phenomenon which are true to some extent and which are useful to take decisions". The set of true assertions represents a model for the phenomenon. Different models for the same phenomenon can exist, which are valid as long as they are applied to take certain kinds of decisions. Consider for instance the Earth model given by the most common

cartography. This model approximates pieces of the globe as it was flat. This is a useful model to trace the shortest route between two points which are not too far from each other, however in an Atlantic crossing it would introduce a non negligible error, and the Earth sphericity should be taken into account. A similar situation applies to our approach. The experiments reported showed that, in an evidence-based methodology to support the user in a classification KDD process, an accurate model of the decision border is not needed to understand and to pre-process the input data. We obtained interesting experimental results for data selection, feature reduction and for the selection of the kernel parameter of the SVM algorithm even with a rough model of the classification problem. However, it is clear that, at least for the definition of the correct classification rule an accurate decision border is needed. Also, in order to build up a valid knowledge-based registry, the decision border description should guarantee an good accuracy. We plan to deeply analyze the issue of decision border accuracy in future works.

Acknowledgements

We thanks the anonymous reviewers that helped with their suggestions to improve the final quality of this manuscript.

This work is dedicated to the memory of our dear co-author Maurizio Panti, who passed away during the last revision of the paper. His vision of semantics in the KDD field deeply inspired and guided our research.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. Brazdil, P., Soares, C. and Costa, J. Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Machine Learning*, 50(3):251–277, 2003.
3. Cannataro, M. and Comito, C. A Data Mining Ontology for Grid Programming. In *Proc. 1st Work. on Semantics in Peer-to-Peer and Grid Computing*, pages 119–130, 2003.
4. Cannataro, M. and Talia, D. The Knowledge Grid. *Comm. of the ACM*, 46(1):89–93, Jan. 2003.
5. Cespivova, H., Rauch, J., Svatek, V., Kejkula, M. and Tomeckova, M. Roles of Medical Ontologies in Association Mining CRISP-DM Cycle. In *ECML/PKDD Workshop on Knowledge Discovery and Ontologies*, pages 1–12, Pisa, Italy, 2004.
6. Chervenak, A., Foster, I., Kesselman, C. and Tuecke, S. Protocols and Services for Distributed Data-Intensive Science. In *Proc. Advanced Computing and Analysis Techniques in Physics (ACAT2000)*, pages 161–163, 2000.
7. Clarkson, K. A program for convex hulls. http://cm.bell-labs.com/netlib/voronoi/hull.html.
8. Diamantini, C. and Spalvieri, A. Quantizing for Minimum Average Misclassification Risk. *IEEE Trans. on Neural Networks*, 9(1):174–182, Jan. 1998.

9. Diamantini, C., Potena, D. and Panti, M. Developing an Open Knowledge Discovery Support System for a Network Environment. In *Proc. of the 2005 International Symposium on Collaborative Technologies and Systems*, page to appear, Saint Louis, Missouri, USA, May 15-19 2005.
10. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
11. Fermandez, C., Martinez, J.F., Wasilewska, A., Hadjimichael, M. and Menasalvas, E. Data Mining - a Semantic Model. In *IEEE International Conference on Fuzzy Systems*, volume 2, pages 938–943, May 2002.
12. Robert Grossman, editor. *Proc. of the Second Annual ACM KDD Workshop on Data Mining Standards, Services and Platforms*, Seattle, WA, Aug. 2004.
13. Grossman, R. and Mazzucco, M. DataSpace: a Data Web for the Exploratory Analysis and Mining of Data. *IEEE Computing in Science and Engineering*, 4(4):44–51, July-Aug. 2002.
14. Grossman, R., Hornik, M. and Meyer, G. Emerging Standards and Interfaces in Data Mining. In Nong Ye, editor, *Handbook of Data Mining*. Kluwer Ac. Pub., Apr. 2003.
15. Hart, P. E. The Condensed Nearest Neighbor Rule. *IEEE Trans. on Information Theory*, 14:515–516, 1968.
16. Hotho, A., Staab, S. and Stumme, G. Ontologies Improve Text Document Clustering. In *IEEE International Conference on Data Mining*, pages 541–544, Nov. 2003.
17. Kalousis, A. and Hilario, M. Model Selection via Meta-Learning. *Int. Journal on Artificial Intelligence Tools*, 10(4), 2001.
18. Kargupta, H., Joshi, A., Sivakumar, K. and Yesha, Y. *Data Mining, Next Generation Challenges and Future Directions*. AAAI/MIT Press, 2004.
19. Kohonen, T., Barna, G. and Chrisley, R. Statistical Pattern Recognition With Neural Networks: Benchmarking Studies. In *IEEE International Conference on Neural Networks*, pages 61–68, San Diego CA, 24-27 Jul 1998.
20. Kotasek, P. and Zendulka, J. An XML Framework Proposal for Knowledge Discovery in Databases. In *European Conference on Principles and Practice of Knowledge Discovery in Databases, Workshop on Knowledge Management: Theory and Applications*, pages 143–156, Lyon, France, 2000.
21. Krishnaswamy, S., Zaslasvky, A., and Loke, S, W. Internet Delivery of Distributed Data Mining Services: Architectures, Issues and Prospects. In V.K. Murthy and N. Shi, editors, *Architectural Issues of Web-enabled Electronic Business*, chapter 7, pages 113–127. Idea Group Publishing, 2003.
22. Kumar, A. Kantardzic, M., Ramaswamy, P. and Sadeghian, P. An Extensible Service Oriented Distributed Data Mining Framework. In *Proc. IEEE/ACM Intl. Conf. on Machine Learning and Applications*, Louisville, KY, USA, 16-18 Dec. 2004.
23. Lee, C. and Landgrebe, D.A. Feature Extraction Based on Decision Boundaries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):288–400, April 1993.
24. Morgera, S.D. and Datta , L. Towards a Fundamental Theory of Optimal Feature Selection: Part I. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(5):601–616, Sept. 1984.
25. Phillips, J. and Buchanan, B.G. Ontology-Guided Knowledge Discovery in Databases. In *1st ACM Int. Conf. on Knowledge Capture*, pages 123–130, Victoria, Canada, October 2001.

26. Sarawagi, S. and Nagaralu, S.H. Data Mining Models as Services on the Internet. *ACM SIGKDD Explorations*, 2(1):24–28, June 2000.
27. Shearer, C. The CRISP-DM Model: The new Blueprint for Data Mining. *Jour. of Data Warehousing*, 5(4), Fall 2000.
28. Talia, D. The Open Grid Services Architecture: Where the Grid Meets the Web. *IEEE Internet Computing*, 6(6):67–71, Nov/Dec 2002.
29. V. Vapnik. *Statistical Learning Theory*. J. Wiley and Sons, New York, 1998.
30. Varde, A., Rundensteiner, E., Ruiz, C., Maniruzzaman, M. and Sisson, R. Data Mining Over Graphical Results of Experiments With Domain Semantics. In *ACM 2nd Internationa Conference on Intelligent Computing and Information Systems*, Cairo, Egypt, 5-7 March 2005.
31. Verschelde, J., Casella Dos Santos, M., Deray, T., Smith, B. and Ceusters, W. Ontology-Assisted Database Integration to Support Natural Language Processing and Biomedical Data Mining. *Journal of Integrative Bioinformatics*, Jan. 2004.
32. Wang, B., McKay, R., Abbass, H. and Barlow, M. A Comparative Study for Domain Ontology Guided Feature Extraction. In *26th Australasian Computer Science Conference*, pages 69–78, Adelaide, Australia, 2003.
33. Yuhua Li and Zhengding Lu. Ontology-Based Universal Knowledge Grid: Enabling Knowledge Discovery and Integration on the Grid. In *IEEE International Conference on Services Computing*, pages 557–560, Sept. 2004.

A Algorithm for the Calculus of Voronoi Diagrams

The Appendix illustrates the algorithm for the analytical description of a Voronoi diagram, starting from the set of code vectors in general dimensional spaces $\mathcal{R}^n$. The algorithm is based on the notion of *Delaunay triangulation*.

Definition A-1. *Given a set of points S in $\mathcal{R}^2$, the* convex hull *of S is the smallest convex set in $\mathcal{R}^2$ containing S.*

Definition A-2. *Given a set of points S in $\mathcal{R}^2$, and its convex hull $\mathcal{H}$, the* Delaunay triangulation *of S is defined as the the unique triangulation of $\mathcal{H}$ such that the points in S are the vertices of the triangles, and no point of S falls inside the triangle's circumcircle.*

The definition of Delaunay triangulation can be extended to any space $\mathcal{R}^n$. In this case, it is also called *Complex of Delaunay*, and triangles are called *simplices*. The circumscribing spheres are called *circospheres*, and their centers are named *circumcenters*.

The following algorithm, starting from the Complex of Delaunay, allows to calculate the Voronoi diagram of the convex hull of a given set S.

Let be given the set $S = S_1 \cup S_2$, where $S_1 = \{m_1, m_2, \ldots, m_M\} \subseteq \mathcal{R}^n$ and $S_2 = \{m_{M+1}, m_{M+2}, \ldots, m_{M+2^n}\} \subseteq \mathcal{R}^n$, where the elements of S_2 are the vertices of an hypercube Φ, such that $m_i \in \Phi$ and the side of $\Phi \gg \| m_i - m_j \|^2$, $\forall i, j \in [1, M]$.

1. Calculate all the hyperplanes

$$\mathcal{S}_{i,j} : \| x - m_i \|^2 = \| x - m_j \|^2, i, j = 1, \ldots, M$$

2. Extract all the $\binom{M+2^n}{n+1}$ permutations of $n+1$ points belonging to S[1];

3. For each permutation $P_i = \{m_{Pi_1}, \ldots, m_{Pi_{n+1}}\}$, if its elements satisfy the definition A-2, calculate the circumcenter C_{P_i} and the radius r_{P_i} of the circosphere. Let H be the number of different circospheres found;

4. For each point $m_i \in S$:
   (a) The vertices of the Voronoi region $\mathcal{V}_i$ are all the circumcenters C_h, $h = 1, \ldots, H$, such that: $\| C_h - m_i \|^2 = {r_h}^2$;

   (b) Extract all the $\binom{H}{n}$ permutations of n vertices of $\mathcal{V}_i$; [2]

[1] Notice that $n+1$ points in $\mathcal{R}^n$ univocally define a simplex.
[2] The choice of the set S guarantees that n such circumcenters always exist.

(c) For each permutation $Q_i = \{C_{Qi_1}, \ldots, C_{Qi_n}\}$, calculate the hyperplane including the points $C_{Qi_1}, \ldots, C_{Qi_n}$. If this hyperplane belongs to the set of hyperplanes generated in step 1, this is a piece of the border of the $\mathcal{V}_i$ region, that is bounded by the circumcenters in Q_i.

Notice that we define the set S of input points as the union of two sets: S_1, which represents the code vectors of the VQ we want calculate the Voronoi diagram of, and S_2, which represents the 2^n vertices of a hypercube containing the real code vectors and having the side much wider than the maximal dimension of the convex hull of S_1. These points represent "points at infinite" and are named *fictitious code vectors*. In this way, the convex hull of S is an expansion of the convex hull of S_1 and the outcome diagram contains all the Voronoi borders of S_1. However, by introducing S_2 we get also fictitious circumcenters and fictitious hyperplanes. It is not difficult to individuate and eliminate fictitious circumcenters and hyperplanes: the former are represented by values which are closer to the values of fictitious code vectors than to real code vectors. The latter are identified as the hyperplanes separating the Voronoi regions of two code vectors at least on of which is a fictitious code vector. Finally, notice that the result is affected by an error which depends on the distance between the elements of S_2 and S_1. For these reasons, it is important that fictitious code vectors are chosen to be very far from the real code vectors.