Sequential Labeling with Structural SVM Under Non-decomposable Loss Functions

Guopeng Zhang

Supervisor: Prof. Massimo Piccardi

Faculty of Engineering and Information Technology
University of Technology, Sydney

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2017

Declaration

I, Guopeng Zhang, hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Guopeng Zhang
March 2017

Acknowledgements

It is not easy to just say "I would like to thank someone" in the acknowledgements section of a PhD thesis. And I really lack words to express my heartfelt thanks to those who have helped me during my PhD study. However, I will always remember that I had the privilege to work with such excellent researchers.

- I would like to sincerely thank Prof. Massimo Piccardi for his kind help during the whole period of my PhD study. Prof. Massimo Piccardi constantly held meetings with me every week to give essential advice on theory and paper annotation in order to help me become a researcher. He is not only a professor but a good friend to me. Since I first came to Sydney he has cared about my life in every respect. It has been an honor for me to be his student and to work with him.
- Dr. Ava Bargi and Dr. Ehsan Zare Borzeshi are also very good researchers. They both started PhD study earlier than me and gave me helpful advice. We frequently exchanged ideas and held discussions with each other to improve our paper writing. I am very glad to have had the chance to work with them.
- I would also like to thank Dr. Shaukat Abidi, Mr. Zhen Wang, Mrs. Fairouz Hussein and Mr. Sari Awwad. Although PhD study is hard and tiring, I was always happy to chat and drink coffee with you all!
- I would like to dedicate this thesis to my loving parents Zhenbo Zhang and Meilin Zhou who supported me emotionally and financially for the duration of my time in Sydney. They made every effort to ensure that I studied and lived well here.
- Last but not least, I would like to extend my sincere thanks to my lovely wife Zhibin Li, who has been my constant support for over 11 years. She has given me more than I can ever say. She worked hard to earn money during my study and cooked meals for me. She encouraged me whenever I felt down or fatigued Without her, I definitely would not have got through the years of exacting study and research. I thank her most

sincerely for everything that she has done for me.

Guopeng Zhang,
June 2016, Sydney

Abstract

This thesis mainly focuses on the sequential labeling problem. Sequential labeling is a fundamental problem in computer vision and machine learning areas and has been researched in many applications. The most popular model for sequential labeling is the hidden Markov model where the sequence of class labels to be predicted is encoded as a Markov chain. In recent years, other structural models, in particular, the extension of SVM to the classification of sequences and other structures have benefited from minimum-loss training approaches which in many cases lead to greater classification accuracy. However, SVM training requires the choice of a suitable loss function. Common loss functions available for training are restricted to decomposable cases such as the zero-one loss and the Hamming loss. Other useful losses such as the F_1 loss, average precision (AP) loss, equal error rates and others are not available for sequential labeling. For the average precision, some results have been proposed in the past, but our results are more general. On the other hand, classification accuracy often suffers from the uncertainty of ground truth labeling and traditional structural SVM only ensures that the ground-truth labeling of each sample receives a score higher than that of any other labeling. However, no specific score ranking is imposed among the other labelings.

For the loss functions problem, we propose a training algorithm that can cater for the F_1 loss and any other loss function based on the contingency table. In our thesis, we propose exact solutions for the F_1 loss, precision/recall at fixed value of recall/precision, precision for a fixed value of predicted positives ("precision at k"), precision/recall Break-Even Point and a formulation of the Average Precision (AP loss). For further experiments, we not only apply the AP loss in the training, but also in testing.

For the uncertainty in the ground-truth labeling problem, we extend the standard constraint set of structural SVM with constraints between "almost-correct" labelings and less desirable ones to obtain a partial ranking structural SVM (PR-SSVM) approach.

We choose different datasets to verify our approaches: human activity datasets including the challenging TUM Kitchen dataset and CMU-MMAC dataset, and the Ozone Level Detection dataset. The experimental results show the efficiency of our approaches on different performance measurements, such as detection rate, false alarm rate and F_1 measure, compared to the conventional SVM, HMM and structural SVM with decomposable losses such as the 0-1 loss and Hamming loss.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Human activity recognition is one of the most important and interesting research areas in computer vision and pattern recognition. The formal goal of human activity recognition is to assign the class labels to images and video frames, no matter if they are for online and offline interactive applications. Multi-class classification is common and human activities are often performed in a time sequential manner.

One of the most interesting and sophisticated research problems for human activity recognition is **Sequential labeling**, also known as tagging or decoding, which assigns a sequence of class labels to a sequence of measurements. Sequential labeling is a fundamental problem in many image and video processing applications such as image segmentation, handwriting recognition, pose estimation, and also in fields as diverse as gene finding, protein classification, natural language processing and financial analysis [88, 46, 25, 59].

Many research works use support vector machine (SVM) for classification as SVM is trained by minimising the regularised empirical loss over a given training set and often achieves a solid reputation for accuracy. However, this kind of approach only considers the training instances as independent and identically distributed (i.i.d) and totally ignores the interdependent relations between instances, which are instead important in sequential labeling or human activity recognition.

For sequential classification, the most popular model is undoubtedly the hidden Markov model (HMM) [107]. The parameters of HMM have traditionally been learned by maximising the likelihood function over a set of supervised (or unsupervised) examples [107]. However, in recent years, other styles of training have become increasingly popular, in particular, the extension of SVM to the classification of sequences and other structures [127, 134].

However, SVM training requires the choice of a suitable loss function. Common loss functions include the misclassification rate (also known as the 0-1 loss), the Hamming loss, the F_1 loss, the precision loss at a given value of recall, the recall loss at a given value of precision, the precison/recall break-even point, the average precision and several others. Depending on different applications, certain loss functions are more suitable than others in providing a desirable classification performance. For instance, in the case of sequences with very sparse positive labels (such as in event or activity detection), a loss function striking a balance between precision and recall such as the F_1 loss is certainly more meaningful than the simple misclassification rate. We will address various loss functions later.

The original formulation of the SVM minimises the 0-1 loss [21]. However, Joachims in [56] has proposed an algorithm for training an SVM under different loss functions including the F_1 loss and all loss functions that can be computed from the classification contingency table (the table with the counts of true positives, true negatives, false positives and false negatives). Similarly, [56] has provided an algorithm of linear complexity for training SVM by minimising the ROC area loss. Yue *et al.* in [166] have provided an algorithm for training SVM under a relaxation of the average precision. Mao and Tsang in [82] have presented an approach for sparse feature selection with SVM under various loss functions. However, all these approaches have only considered independent and identically distributed (i.i.d.) data and very little work has been done to date on addressing training under various losses for sequential labeling and structured prediction. In the original structural SVM (SSVM) papers [133, 134], Tsochantaridis *et al.* have hinted at the possibility of training SVM for sequential labeling under other losses, but they have not provided explicit algorithms. In [118], Rosenfeld *et al.* have proposed an algorithm for training SSVM under the average precision loss. However, the approach requires that all the samples in the training set be manually ranked in a global order relationship by a ground-truth annotator. Since this is generally not feasible, the authors treat the ranking as a latent variable and introduce approximations for both training and inference.

In this thesis, we extend the work from i.i.d. data to sequential data to present an approach for training SSVM for sequential labeling under a number of loss functions, including the F_1 loss, the precision (recall) at a set point of recall (precision), the precision/recall break-even point and any other function based on the classification contingency table. In addition, we present an algorithm for training SSVM under a formulation of the average precision, and we also use this algorithm in inference.

Furthermore, we consider cases such as human activity segmentation where the manual annotation of the start and end of an activity carries a significant degree of uncertainty. We wish to ensure that the ground truth labeling receives the highest score, meanwhile labelings which are close to the ground truth receive a higher score than other less qualified labelings. In other word, "partial ranking" is needed in our proposed technique. So we refer to the proposed method as partial-ranking structural SVM.

All experimental results conducted on different sequential datasets including the challenging TUM kitchen mocap dataset, the CMU-MMAC video dataset and the Ozone Level Detection dataset in terms of F_1 measure, detection rate and false alarm rate have shown that our proposed methods outperform other training approaches such as SVM, HMM and conventional structural SVM with the 0-1 loss and the Hamming loss.

Following this brief introduction, Chapter 2 gives a more formal literature review of activity recognition, multi-label classification, sequential labeling and SVM, including binary SVM, multi-class SVM, structural SVM and latent structural SVM. Chapter 3 presents a training algorithm that can cater for the F_1 loss and any other loss function based on the contingency table. Chapter 4 presents an algorithm for training structured models with any non-decomposable loss and the average precision loss. Chapter 5 extends the standard constraint set of structural SVM with constraints between "almost-correct" labelings and less desirable ones to obtain a partial ranking structural SVM (PR-SSVM) approach. Chapter 6 extends the AP loss not only for training but also for the inference procedure. The conclusion is given in Chapter 7.

Published work can be found as follows:

- Sequential Labeling With a Binary-state HMM Under the F_1 loss. Guopeng Zhang & Massimo Piccardi. In Proceedings of the 21st IEEE International Conference on Image Processing (ICIP) 2014, pages 5272-5276.

- Structural SVM With Partial Ranking for Activity Segmentation and Classification. Guopeng Zhang & Massimo Piccardi. Published in IEEE Signal Processing Letters (Volume 22, December 2015), pages 2344-2348.

- Sequential Labeling With Structural SVM Under Non-decomposable Losses. Guopeng Zhang & Massimo Piccardi. Submitted to the IEEE Transactions on Neural Networks and Learning Systems, August 2015

- Sequential Labeling With Structural SVM Under the Average Precision Loss. Guopeng Zhang & Massimo Piccardi. Submitted to the joint IAPR International Workshops on Structural and Syntactic Pattern recognition (SSPR 2016) and Statistical techniques in Pattern Recognition (SPR 2016).

Chapter 2

Literature Review

In this chapter, we review the fundamental background of human activity segmentation and classification, including activity recognition and machine learning techniques, multi-label classification, sequential labeling, support vector machines (binary SVM, multi-class SVM, structural SVM and latent structural SVM).

2.1 Activity Recognition Review

2.1.1 Taxonomy of Human Activities

The problem of recognizing human activities, such as *jogging*, *boxing*, *walking*, *running*, *fighting* from video sequences has drawn increasing attention in terms of research. Human activity analysis has become one of the most popular research areas in computer vision and many other research areas. Many applications for human activity recognition have been found in areas such as visual surveillance, human performance analysis, computer-human interfaces (robotic interaction with humans), content-based image retrieval/storage and virtual reality.

The interest in the topic is motivated by the promise of many applications, both online and offline. Automatic annotation of video enables more efficient searching, for example finding tackles in soccer matches, handshakes in news footage or typical dance music videos. Online processing allows for automatic surveillance, for example in shopping malls, but also in smart homes for the elderly to support aging at home. Interactive applications, for example in human-computer interaction or games, also benefit from the advances in automatic human action recognition.

There have been several existing surveys in the area of vision concerning human motion analysis and recognition. Recent overviews by Forsyth et al [36] and Poppe [103] focus on the recovery of human poses and motion from image sequences. This problem is actually a regression problem, whereas human activity recognition is a classification problem. However, there are many similarities within these two topics, especially at the level of image representation. The work on human/pedestrian detection is also related, where the task is to localise persons within the image.

Bobick proposed a taxonomy of movement recognition, activity recognition and action recognition. Basically, these three classes correspond with low-level, mid-level and high-level vision tasks [11]. Aggarwal and Cai [55], and later Wang et al. [80], did research on body structure analysis, tracking and recognition. Gavrila used taxonomy of 2D approaches, 3D approaches and recognition [40]. Moeslund et al. used a functional taxonomy in subsequent phases: initialization, tracking, pose estimation and recognition [129]. Within the recognition task, different parts such as scene interpretation, holistic approaches, body-part approaches and action primitives were discussed. Turaga et al. in a recent survey focused on the higher-level recognition of human activities [106]. Furthermore, Kruger et al. discussed intention recognition and imitation learning in [142].

2.1.2 Activity Segmentation and Classification

There are two commonly considered problems in human activity recognition. The first one is called segmentation and the other one is classification. Segmentation is commonly used in tasks such as image, video and speech segmentation. The problem of segmentation is to consider the partitioning of a digital image into two or more regions (image segmentation) or dividing the video sequence into proper segments of frames (video segmentation). The classification problem is also related to categorization which is the process by which objects are recognized, differentiated and understood. Each object will be assigned a class labeling in the classification process.

In human activity recognition, the common approach is to extract image features from the video and then assign a corresponding class label. The classification algorithm is usually learned from training data. In this section, we discuss the challenges that influence the choice of image representation and classification algorithm.

There are many difficulties in terms of this procedure. For many activities, different instances will differ from each other. For example, walking movements look different because of the different speed and stride length in different environments. Also, there are anthropometric differences between individuals. Similar observations can be made for other activities, especially for non-cyclic activities or activities that are adapted to the environment (e.g. avoiding obstacles while walking, or pointing towards a certain location). A good human activity recognition approach should generalise well over variations within one class and distinguish between activities of different classes. For increasing numbers of actions classes, this will be more challenging as the overlap between classes will be higher. In some domains, a distribution over class labels might be a suitable alternative.

The environment in which the activity performance takes place causes variation in the recording. Person localisation might prove harder in cluttered or dynamic environments. Moreover, parts of the person might be occluded in the recording. Lighting conditions can further influence the appearance of the person. The same activities observed from different viewpoints can lead to very different image observations. Assuming a known camera viewpoint restricts the use to static cameras. When multiple cameras are used, the viewpoint problem is essential, especially when observations from multiple views can be combined into a consistent representation. Dynamic background makes the complexity of localising the person in the image increase sharply. When using a moving camera, these challenges will become even harder. In vision-based human activity recognition, all these issues should be addressed explicitly.

2.1.3 Commonly Used Datasets in Human Activity Recognition

Many researchers use publicly available datasets that are specifically recorded for training and evaluation in their work. This provides a sound mechanism for comparison but the datasets often lack some of the earlier mentioned variations.

In recent years, more realistic datasets have been used for research. These datasets contain labelled sequences gathered from movies or web videos. Although these datasets can avoid some specific problems, they are still limited in the number of training and test sequences. Labeling these sequences is also a challenge. Several automatic approaches have been proposed, for example using web image search results, video subtitles and subtitle to movie script matching. Gadison et al. [39] presented an approach to re-rank automatically extracted and aligned movie samples, but manual verification is usually necessary [43]. Also, performance of an activity might be perceived differently. Significant disagreement has been found by a small-scale experiment between human labeling and the assumed ground-truth on a common dataset [99]. When no labels are available, an unsupervised approach needs to be used but there is no guarantee that the discovered classes are meaningful.

Amongst the most commonly used dataset, we mention: 1) *KTH human motion dataset* [121]. The KTH human motion dataset contains six actions (walking, jogging, running, boxing, hand waving and hand clapping), performed by 25 different actors. There are 2391 video samples in total. Four different scenarios are used: outdoors, outdoors with zooming, outdoors with different clothing and indoors. There are a lot of variations in the performance and duration, and also in the viewpoint (Fig. 2.1). The backgrounds are relatively static.

Apart from the zooming scenario, there is only slight camera movement. KTH uses average accuracy over all classes as the performance measure.

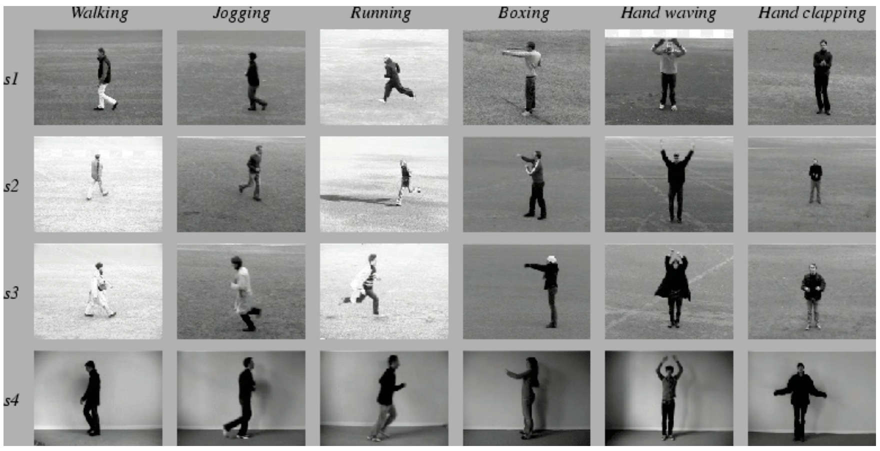


Fig. 2.1 Different actions contained by KTH human motion dataset.

2)*Weizmann human action dataset* [10]. The human action dataset recorded at the Weizmann institute contains 10 actions (walking, running, jumping, galloping sideways, bending, one-hand waving, two-hand waving, jumping in place, jumping jack and skipping), each performed by 10 people (Fig. 2.2). The backgrounds are static and foreground silhouettes are included in the dataset. The viewpoint is static. In addition to this dataset, two separate sets of sequences were recorded for robustness evaluation. One set shows walking movement viewed from different angles. The second set shows fronto-parallel walking actions with slight variations (carrying objects, different clothing, different styles).

3) *UCF sports action dataset* [115]. The UCF sports actions dataset contains 150 sequences of sport motions (diving, golf swinging, kicking, weightlifting, horseback riding, running, skating, swinging a baseball bat and walking) (Fig. 2.3). Bounding boxes of the human figure are provided with the dataset. For most action classes, there is considerable variation in action performance, human appearance, camera movement, viewpoint, illumination and background. The evaluation method is leave-one-out. It also uses average accuracy over all classes as its performance measure.

4) *Hollywood human action dataset (HoHA)* [71]. The Hollywood human action dataset contains eight actions (answering phone, getting out of car, handshaking, hugging, kissing, sitting down, sitting up and standing up), extracted from movies and performed by a variety of actors (Fig. 2.4). A second version of the dataset (HoHA2) includes four additional actions (driving car, eating, fighting, running) and an increasing number of samples for each class. One training set is automatically annotated using scripts of the movies, another is manually

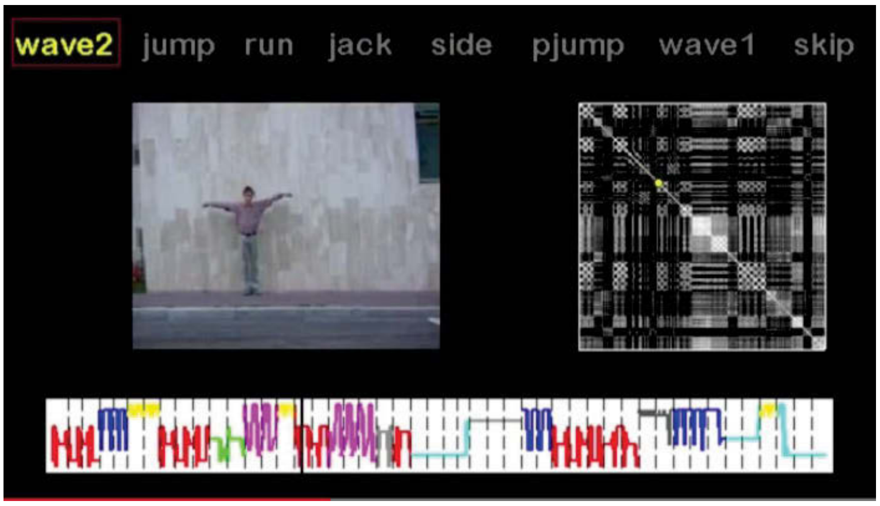


Fig. 2.2 Human action dataset recorded at the Weizmann institute.

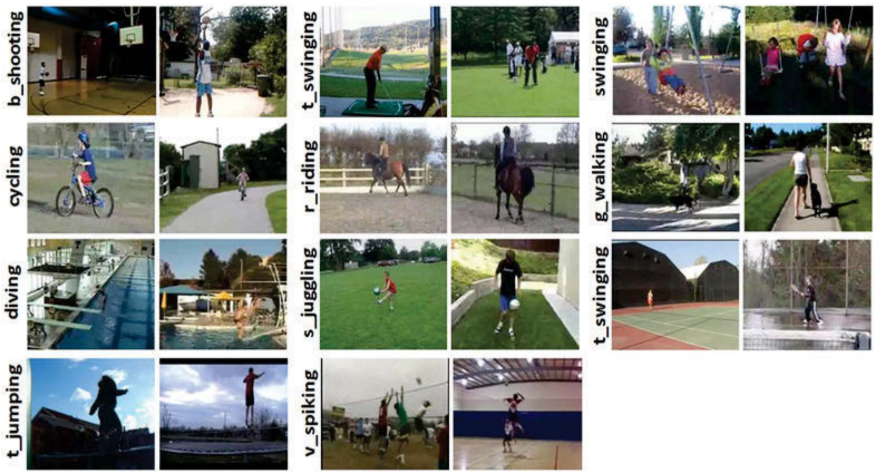


Fig. 2.3 UCF sports action dataset.

labelled. There is a huge variety of performance of the actions, both spatially and temporally. Occlusions, camera movements and dynamic backgrounds make this dataset challenging. Most of the samples are at the scale of the upper-body but some show the entire body or a close-up of the face.

5) *INRIA XMAS multi-view dataset* [156]. Weinland et al. introduced the IXMAS dataset that contains actions captured from five viewpoints. A total of 11 people perform 14 actions (checking watch, crossing arms, scratching head, sitting down, getting up, turning around, walking, waving, punching, kicking, pointing, picking up, throwing over head and throwing from bottom up). The actions are performed in an arbitrary direction with regard

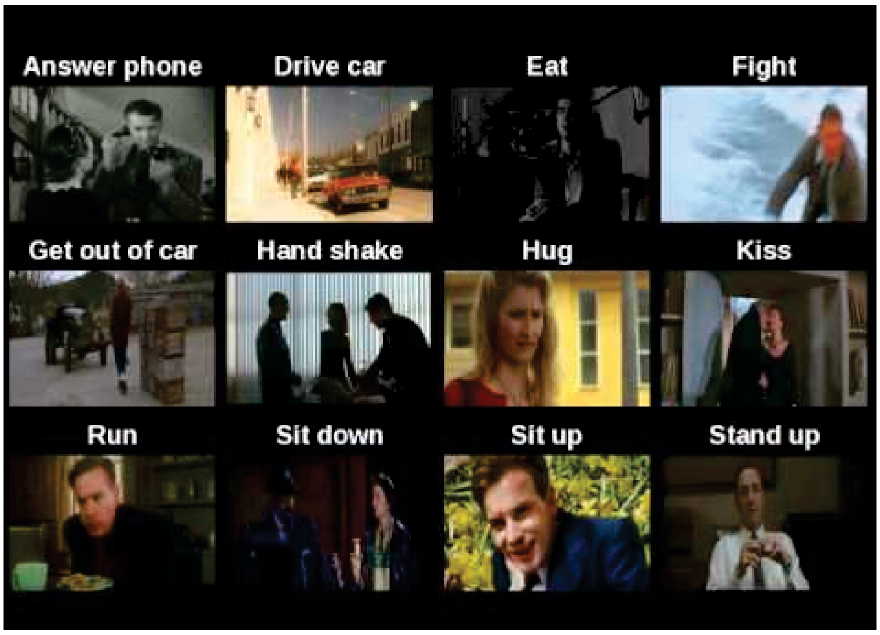


Fig. 2.4 Hollywood human action dataset.

to the camera setup (Fig. 2.5). The camera views are fixed, with a static background and illumination setting. Silhouettes and volumetric voxel representations are part of the dataset.

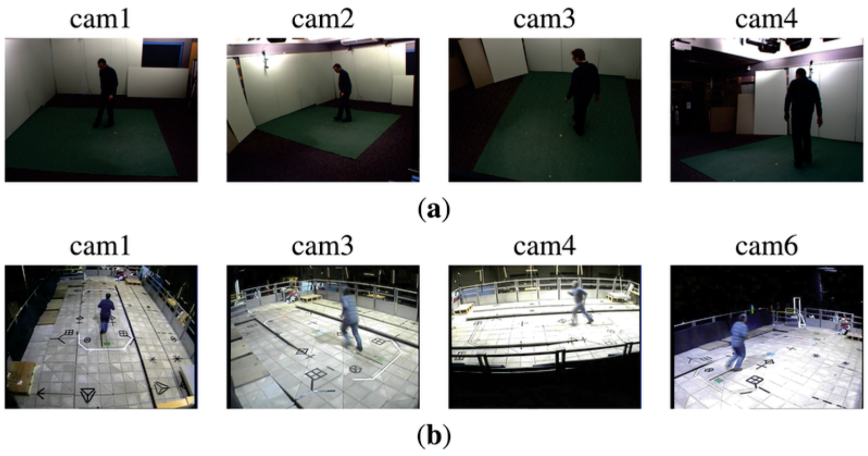


Fig. 2.5 INRIA XMAS multi-view dataset.

6) *TUM Kitchen dataset* [130]. The TUM Kitchen dataset is a collection of activity sequences recorded in a kitchen environment equipped with multiple sensors [130]. In the kitchen environment, various human subjects were asked to set a table in different ways, performing 9 actions, namely *Reaching*, *Carrying*, *TakingSomething*, *LoweringAnObjec-*

t, *ReleasingGrasp*, *OpeningADoor*, *ClosingADoor*, *OpeningADrawer*, *ClosingADrawer* (Fig. 2.6). The dataset contains multiple, simultaneous types of data: video data from 4 fixed overhead cameras, motion capture data extracted from the videos, RIFD tag readings from 3 fixed readers embedded in the environment, magnetic sensor readings from objects and the environment, and their annotated action labels.

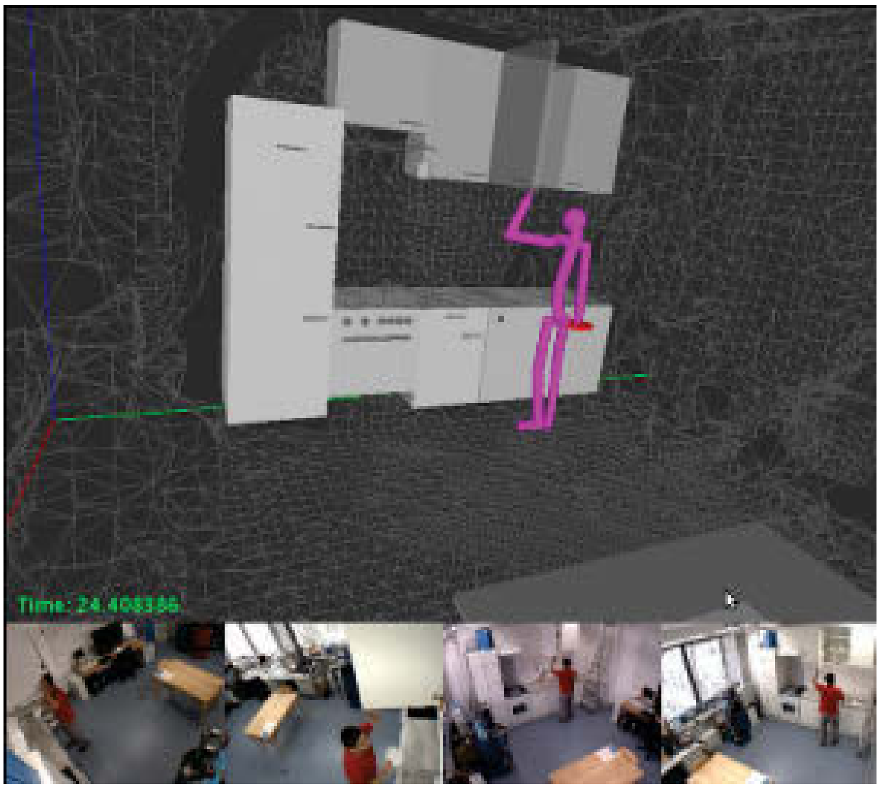


Fig. 2.6 TUM kitchen dataset.

7) *CMU-MMAC dataset* [24]. The CMU-MMAC dataset was collected at Carnegie Mellon University's Motion Capture Lab and contains multimodal measurements of the human activity of 55 subjects preparing 5 different recipes: brownies, salad, pizza, sandwich and scramble eggs [24] (Fig. 2.7). The recorded modalities include: a) Video: (1) Three high spatial resolution (1024×768) color video cameras at low temporal resolution (30 Hertz). (2) Two low spatial resolution (640×480) color video cameras at high temporal resolution (60 Hertz). (3) One wearable low high spatial resolution ($800 \times 600/1024 \times 768$) camera at low temporal resolution (30 Hertz). b) Audio: Five balanced microphones. c) Motion Capture: A Vicon motion capture system with 12 infrared MX-40 cameras. Each camera records images at 4 megapixel resolution at 120 Hertz. d) Internal Measurement Units (IMUs): (1) Wired

IMUs (3DMGX). (2) Bluetooth IMUs(6DOF). e) Wearable devices: (1) BodayMedia. (2) eWatch.

Fig. 2.7 CMU MMAC dataset.

8) *Ozone Level Detection dataset* [168]. Although the Ozone Level Detection dataset is not a human activity dataset, it is an event detection sequential dataset that can be used for sequential labeling. Therefore, we have also used this dataset in our experiments.

The Ozone Level Detection dataset is a sequential binary dataset (ozone day and non ozone day)that was collected from 1/1/1998 to 31/12/2004 in the Houston, Galveston and Brazoria areas, containing 2536 instances with 72 features for each instance [168]. The dataset is sparse (only about 5% positive depending on the criteria of "ozone days") and evolves over time from year to year containing a large number of irrelevant features. There are two annotations for the collection, one is the "eight hour" peak and the other is the "one hour" peak.

2.1.4 Image Representation

Ideally, these should generalize over small variations in personal appearance, background, viewpoint and activity execution. At the same time, the representations must be sufficiently rich to allow for robust classification of the activity. The temporal aspect is important in activity performance. Some of the image representations explicitly take into account the temporal dimension, others extract image features for each frame in the sequence individually. In this case, the temporal variations need to be dealt with in the classification step.

Image representations can be divided into two categories: global representations and local representations. The former encodes this visual observation as a whole. Global representations are obtained in a top-down fashion: a person is localized first in the image using background subtraction or tracking. Then, the region of interest is encoded as a whole, which results in the image descriptor. The representations are powerful since they encode much of the information. However, they rely on accurate localization, background subtraction or tracking. Also, they are more sensitive to viewpoint, noise and occlusions. When the domain allows for good control of these factors, global representations usually perform well.

Local representation describes the observation as a collection of independent patches. The calculation of local representations proceeds in a bottom-up fashion: spatio-temporal interest points are detected first, and local patches are calculated around these points. Finally, the patches are combined into a final representation. After the initial success of bag-of-feature approaches, there is currently more focus on the correlations between patches. Local representations are less sensitive to noise and partial occlusion, and do not strictly require background subtraction or tracking. However, as they depend on the extraction of a sufficient amount of relevant interest points, pre-processing is sometimes needed, for example to compensate for camera movements.

Global Representations

Global representations encode the regions of interest (ROI) of a person as a whole. The ROI is usually obtained through background subtraction or tracking. Common global representations are derived from silhouettes, edges or optical flow. They are sensitive to noise, partial occlusions and variations in viewpoint. To partly overcome these issues, grid-based approaches spatially divide the observation into cells, each of which encodes part of the observation locally. Multiple images over time can be stacked, to form a three-dimensional space-time volume, where time is the third dimension. Such volumes can be used for activity recognition.

The silhouette of a person in the image can be obtained by using background subtraction. In general, silhouettes contain some noise due to imperfect extraction. Also, they are somewhat sensitive to different viewpoints, and implicitly encode the anthropometry of the person. Still, they encode a great deal of information. When the silhouette is obtained, there are many different ways to encode either the silhouette area or the contour.

One of the earliest uses of silhouette is by Bobick and Davis [12]. They extract silhouettes from a single view and aggregate differences between subsequent frames of an activity sequence. This results in a binary motion energy image (MEI) which indicates where motion occurs. Also, a motion history image (MHI) is constructed where pixel intensities are a recent function of the silhouette motion. Two templates are compared using Hu moments. Contours are used in [16], where the star skeleton describes the angles between a reference line, and the lines from the center to the gross extremities (head, feet, hands) of the contour. Wang and Suter [144] use both silhouette and contour descriptors. Given a sequence of frames, an average silhouette is formed by calculating the mean intensity over all centered frames. Similarly, the mean shape is formed from the centered contours of all frames. Wenland et al.

[154] match two silhouettes using Euclidean distance. In later work, silhouette templates are matched against edges using Chamfer distance, thus eliminating the need for background subtraction.

When multiple cameras are employed, silhouettes can be obtained from each. Huang and Xu [161] use two orthogonally placed cameras at approximately a similar height and distance to the person. Silhouettes from both cameras are aligned at the medial axis, and an envelope shape is calculated. Cherla et al. [17] also use orthogonally placed cameras and combine features of both. Such representations are somewhat view-invariant, but fail when the arms cannot be distinguished from the body. Weinland et al. [155] combine silhouettes from multiple cameras into a 3D voxel model. Such a representation is informative but requires accurate camera calibration. They use motion history volumes, which is an extension of the MHI [12] to 3D. View-invariant matching is performed by aligning the volumes using Fourier transforms on the cylindrical coordinate system around the medial axis.

Instead of (silhouette) shape, motion information can be used. The observation within the ROI can be described with optical flow, the pixel-wise oriented difference between subsequent frames. Flow can be used when background subtraction cannot be performed. However, dynamic backgrounds can introduce noise in the motion descriptor. Also, camera movement results in observed motion, which can be compensated for by tracking the person. Efros et al. [27] calculate optical flow in person-centered images. They use sports footage, where persons in the image are very small. The result is blurred as optical flow can result in noisy displacement vectors. To make sure that oppositely directly vectors do not even out, the horizontal and vertical components are divided into positively and negatively directed, yielding 4 distinct channels. Ahad et al. [1] use these four flow channels to solve the issue of self-occlusion in a MHI approach. Ali and Sha [2] derive a number of kinematic features from the optical flow. These include divergence, vorticist, symmetry and gradient tensor features. Principal component analysis (PCA) is applied to determine dominant kinematic modes.

Global Grid-based Representations

By dividing the ROI into a fixed spatial or temporal grid, small variations due to noise, partial occlusions and changes in view-point can be partly overcome. Each cell in the grid describes the image observation locally, and the matching function is changed accordingly from global to local. These grid-based representations resemble local representations, but require a global representation of the ROI.

Kellokumpu et al. [64] calculate local binary patterns along the temporal dimension and store a histogram of non-background responses in a spatial grid. Thurau and Hlavac [131] use histograms of oriented gradients (HOG) and focus on foreground edges by applying non-negative matrix factorization. Lu and Little [79] apply PCA after calculating the HOG descriptor, which greatly reduces the dimensionality. Ikizler et al. [49] first extract human poses using [109]. Within the obtained outline, oriented rectangles are detected and stored in a circular histogram. Ragheb et al. [108] transform, for each spatial location, the binary silhouette response over time into the frequency domain. Each cell in the spatial grid contains the mean frequency response of the spatial locations it contains.

Optical flow in a grid-based representation is used by Danafar and Gheissari [23]. They adapt the work of Efros et al. by dividing the ROI into horizontal slices that approximately contain head, body and legs. Zhang et al. [171] use an adaptation of the shape context, where each log-polar bin corresponds to a histogram of motion word frequencies. Combinations of flow and shape descriptors are also common, and overcome the limitations of a single representation. Tran et al. [132] use rectangular grids of silhouettes and flow. Within each cell, a circular grid is used to accumulate the responses. Ikizler et al. [48] combine the work of Efros et al. [27] with histogram of oriented line segments. Flow, in combination with local binary patterns is used in yang2007learning.

Space-time Volumes

A 3D spatio-temporal volume (STV) is formed by stacking frames over a given sequence. Accurate localization, alignment and possibly background subtraction are required.

Blank et al. [10] first stack silhouettes over a given sequence to form a STV. Then the solution of the Poisson equation is used to derive local space-time saliency and orientation features. Global features for a given temporal range are obtained by calculating weighted moments over these local features. To deal with performances of different temporal durations, Achard et al. [1] use a set of space-time volumes for each sequence, each of which covers only a part of the temporal dimension.

Several works sample the STV surface and extract local descriptors. While this approach shares many similarities with local approaches, the STV is a global representation. Batra et al. [6] stack silhouettes, and sample the volume with small 3D binary space-time patches. Yilmaz and Shah [164] use differential geometric properties on the STV surface, such as maxima and minima in the space-time domain. An activity sketch is the set of these local descriptors. The method is sensitive to noise on the surface. The idea is extended by Yan et al. [162] by first constructing 3D exemplars from multiple views, and then calculating an activity sketch from the view-based STV and projecting it on to the constructed 3D exemplars. The activity sketch descriptors encode both shape and motion, and can be matched with observations from arbitrary viewpoints. Grundmann et al. [42] extend the shape context to

3D and apply it to STVs. The sampling of interest points is adapted to give more importance to moving regions.

Jiang and Martin [54] use 3D shape flows over time, calculated at edge points. The matching can deal with cluttered backgrounds. Ke et al. [61] construct an STV of flow and sample the horizontal and vertical components in space-time using a 3D variant of the rectangle features of [140]. Ogata et al. [95] extend this work with [27]. A combination of STVs of silhouettes and flow is used by Ke et al. [63]. No background subtraction is needed, as 3D super-pixels are obtained from segmenting the STV. Activity classification is cast as 3D object matching, where the distance to the segment boundary is used as a similarity measure. The work is extended in [62] to allow for the matching of parts, thus enabling recognition of activities under partial occlusion.

Local Representations

Local representations describe the observation as a collection of local descriptors or patched. Accurate localization and background subtraction are not required and local representations are somewhat invariant to changes in viewpoint, personal appearance and partial occlusions. Patches are sampled either densely or at space-time interest points. The latter are locations that correspond to interesting motions. Local descriptors describe small windows (2D) in an image or cuboids (3D) in a video volume. Similar to global representations, observations can be grouped locally within a grid. By exploiting correlations in space and time between the patches, activities can be modelled more effectively since only the meaningful patches are retained.

Space-time Interest Points Detectors

Space-time interest points are the locations in space and time where sudden changes of movement occur in the video. It is assumed that these locations are most informative for the recognition of human activity. Usually, points that undergo a translational motion in time will not result in the generation of space-time interest points.

Lapyev and Lindeberg [69] extended the Harris corner detector [45] to 3D. Space-time interest points are those points where the local neighbourhood has a significant variation in both the spatial and the temporal domain. The scale of the neighbourhood is automatically selected for space and time on an individual basis. The work is extended to compensate for relative camera motions in [70]. Oikonomopoulos et al. [97] extended the work on 2D salient point detection by Kadir and Brady [60] to 3D. The entropy within each cubiod is calculated, and the centers of those with local maximum energy are selected as salient points. The scale of each salient point is determined by maximizing the entropy values.

One drawback of these method is the relatively small number of stable interest points. This issue is addressed by Dollar et al. [26], who apply Gabor filtering on the spatial and temporal dimensions individually. The number of interest points is adjusted by changing the spatial and temporal size of the neighbourhood in which local minima are selected. Chomat et al. [19] use the responses after applying spatio-temporal receptive fields. In a similar fashion, Rapantzikos et al. [112] apply discrete wavelet transforms in each of the three directions of a video volume. Responses from low-pass and high-pass filtering for each dimension are used to select salient points in space and time. In addition to intensity and motion cues, Rapantzikos et al. [113] also incorporate color. They compute saliency as the solution of an energy minimization process which involves proximity, scale and feature similarity terms.

Willems et al. [158] identify saliency as the determinant of a 3D Hessian matrix, which can be calculated efficiently due to the use of integral videos. Another attempt to decrease the computational complexity is presented by Oshin et al [98], who train randomized ferns to approximate the behavior of interest point detectors. In a comparison on the task of human activity recognition, Wang et al. [143] found that dense sampling outperformed the interest point detectors of Dollar et al. [26], Lapyev and Lindeberg [69], Willems et al. [158].

Instead of detecting interest points over the entire volume, Wong and Cipolla [159] first detect subspaces of correlated movement. These subspaces correspond to large movements such as an arm wave. Within these spaces, a sparse set of interest points is detected. In a similar approach, Bregonzio et al. [14] first calculate the difference between subsequent frames to estimate the focus of attention. Next, Gabor filtering is used to detect salient points within these regions.

Local Descriptors

Local descriptors summarize an image or video patch in a representation that is ideally invariant to background clutter, appearance and occlusions, and possibly to rotation and scale. The spatial and temporal size of a patch is usually determined by the scale of the interest point. Schuldt et al. [121] calculate patches of normalized derivatives in space and time. Niebles et al. [89] take the same approach, but applying smoothing before reducing the dimensionality using PCA. Dollar et al. [26] experiment with both image gradients and optical flow.

Patches can also be described by local grid-based descriptors. These summarized the local observation within grid cells, thus ignoring small spatial and temporal variations. SURF features [7] are extended to 3D by Willems et al. [158]. These eSURF features contain in each cell the sums of Haar-wavelets. Laptev et al. [71] use local HOG and HOF (histogram of oriented flow) descriptors. The extension of HOG to 3D is presented by Klaser et al. [67]. 3D gradients are binned into regular polyhedrons. They extend the idea of integral images into 3D which allows rapid dense sampling of the cuboid over multiple scales and locations in both space and time. In related work by Scovanner et al. [122], the SIFT descriptor [77] is extended to 3D. Wang et al. [143] compared local descriptors and found that, in general, a combination of image gradient and flow information resulted in the best performance.

Several approaches combine interest point detection and the calculation of local descriptors in a feed-forward framework. For example, Jhuang et al. [52] use several stages to ensure invariance to a number of factors. Their approach is motivated by the human visual system. At the lowest level, Gabor filters are applied to dense flow vectors, followed by a local max operation. Then the responses are converted to a higher level using stored prototypes and a global max operation is applied. A second matching stage with prototypes results in the final representation. The work in [91] is similar in concept, but uses different window settings. Schindler and Van Gool [120] extend the work by Jhuang et al. [52] by combining both shape and flow responses. Escobar et al. [29] use motion-sensitive responses and also consider interactions between cells, which allows them to model more complex properties such as motion contrasts.

Comparing sets of local descriptors is not straightforward due to the possibly different number and the usually high dimensionality of the descriptors. Therefore, often a codebook is generated by clustering patches and selecting either cluster centers or the closest patches as codewords. A local descriptor is described as a codeword contribution. A frame or sequence can be represented as a bag-of-words, a histogram of codeword frequencies [89, 121].

Local Grid-based Representations

Similar to holistic approaches, grids can be used to bin the patches spatially or temporally. Compared to the bag-of-words approach, using a grid ensures that spatial information is maintained to some degree.

In the spatial domain, Ikizler and Duygulu [50] sample oriented rectangular patches, which they bin into a grid. Each cell has an associated histogram that represents the distribution of rectangle orientations. Zhao and Elgammal [173] bin local descriptors around interest points in a histogram with different levels of granularity. Patches are weighted according to their temporal distance to the current frame.

Nowozin et al. [92] use a temporal instead of a spatial grid. The cells overlap, which allows them to overcome small variations in performance. Observations are described as PCA-reduced vectors around extracted interest points, mapped onto codebook indices.

Laptev and Perez [72] bin histograms of oriented gradients and flow, extracted at interest points, into a spatio-temporal grid. This grid spans the volume that is determined based on the postilion and size of a detected head. The distribution of these histograms is determined for every spatio-temporal cell in the grid. Three different block types are used to form the new feature set. These types correspond to a single cell, a concatenation of two temporally neighboring cells and a concatenation of spatially neighboring cells. A subset of all possible blocks within the grid is selected using Adaboost. A large number of grid types, with different spatial and temporal divisions and overlap settings ,is evaluated in [71]. Flow descriptors from [27] are used by Fathi and Mori [34], who select a discriminative set of low-level flow features within space-time cells which form an overlapping grid. In a subsequent step, a set of these mid-level features is selected using the AdaBoost algorithm. In the work by Bregonzio et al. [14], no local image descriptors are calculated. Rather, they look at the number of interest points within cells of a spatio-temporal grid with different scales. This approach is computationally efficient but depends on the number and relevancy of the interest points.

Correlations Between Local Descriptors

Grid-based representations model spatial and temporal relations between local descriptors to some extent. However, they are often redundant and contain uninformative features. We describe approaches that exploit correlations between local descriptors for selection or the construction of higher-level descriptors.

Scovanner et al. [122] construct a word co-occurrence matrix, and iteratively merge words with similar co-occurrences until the difference between all pairs of words is above a specified threshold. This leads to a reduced codebook size and similar activities are likely to generate similar distributions of codewords. Similar in concept is the work by Liu et al. [74], who use a combination of the space-time features and spin images, which globally describe an STV. A co-occurrence matrix of the features and the activity videos is constructed. The matrix is decomposed into eigenvectors and subsequently projected onto a lower-dimensional space. This embedding can be seen as feature-level fusion. Instead of determining pairs of correlated codewords, Patron-Perez and Reid [99] approximate the full joint distribution of features using first-order dependencies. Features are binary variables that indicate the presence of a codeword. A maximum spanning tree is formed by analyzing a graph between all pairs of features. The work by Kim et al. [65] is different in the sense that the correlation between two videos is measured. Canonical correlation analysis is extended to handle image sequences. The approach implicitly deals with affine variations. Discriminative features are subsequently selected using Adaboost.

In contrast to the above approaches where spatial information is ignored, Savarese et al. [119] introduce correlations that describe co-occurrences of codewords within spatio-temporal neighbourhoods. The codebook size strongly influences the classification performance. Too few entries do not allow for good discrimination, while too great a codebook size is likely to introduce noise due to sparsity of the histogram. Liu and Shah [76] solve this issue and determine the optimal size of the codebook using maximization of mutual information. This technique merges two codebook entries if they have comparable distributions. In addition, they use spatio-temporal pyramid matching to exploit temporal information. Yao and Zhu [163] introduce an active basis of shape and flow patches, where locations in space and time are allowed to vary slightly.

Correlations between descriptors can also be obtained by tracking features. Sun et al. [126] calculate SIFT descriptors around interest points in each frame and use Markov chaining to determine tracks of these features. Similar work by Messing et al. [85] extracts trajectories using the KLT tracker. In both cases, tracks are summarized in a log-polar histogram of track velocities. Oikonomopoulos et al. [96] fit B-splines to the STV boundary that is formed by a coherent region of saliency responses. Song et al. [125] track points between frames. They fit a triangulated graph to these points to detect and recognize human activities. In Fanti et al. [31], additional local appearance cues are used. Global variables are introduced for scale, viewpoint and translation. These methods assume static background and motion due to objects in the background generating feature tracks that do not belong to the person.

This limitation is partly addressed by Niebles and Fei-Fei [94], who model the frame as a mixture of constellations. Each constellation models the spatial arrangement of codewords instead of tracked features. Filipovych and Ribeiro [35] include both pose constellations and dynamics constellations. Star graphs of static and dynamic features are combined into a tree by conditioning on the landmark vertices of the individual graphs. These models are trained without supervision. Related work introduces hidden variables that correspond to activity categories. Probabilistic latent semantic analysis (pLSA) is a generative model used by Niebles et al. [89]. In an unsupervised way ,the mapping from latent activity labels to distribution of codewords is learned. Wong et al. [160] extend pLSA by introducing the location of a person's centroid. Both works require that the number of activity labels is determined empirically. Instead, Wang et al. [150] take a supervised approach and use a semi-latent Dirichlet allocation (s-LDA) model. In Wang and Mori [151], an adapted hidden conditional random field (HCRF) model is used to learn constellations of codewords discriminatively. A more efficient learning algorithm is presented in [152].

A number of works take the approach of extracting and mining a large number of features. Mikolajczyk and Uemura [86] extract for each frame local shape and motion features. For each feature, the relative location and orientation to a person's center of mass are stored together with the annotated activity label. These features are clustered and represented in vocabulary trees. By matching features extracted from an observed frame, votes are cast over a person's location, orientation and activity label. In Uemura et al. [137], global motion patterns are detected and compensated for, in order to recognize activity from moving cameras. In related work, Gilbert et al. [41] find spatio-temporal corners and determine the relative spatial arrangement of all other corners in the frame. This results in an extremely large number of features. Data mining technique are features which are used to discriminatively select those feature combinations that are informative of a class. Liu et al. [75] select discriminative denatures by applying the PageRank algorithm on the feature co-occurrence graph.

There are relatively few works that address the effect of viewpoint on the recognition of human activities. Farhadi and Tabrizi [32] explicitly address the correlations between activities observed scribe clusters of codewords in each view. The transfer of these splits between views is learned from multi-view activity sequences. Farhadi et al. [33] model the view as a latent parameter, and learn features that can discriminate between views and activities.

2.1.5 Application-specific Representations

In contrast to the more general image representations that have been discussed in the previous sections, a number of works use representations that are directly motivated by the domain of human activity recognition.

Joint locations or joint angles are rich representations, but it is challenging to derive them from video. In 3D, the representations are completely view-invariant, whereas for 2D, there have been several approaches proposed to address the issue of matching 2D joint trajectories to activity labels. Since we focus on the recognition of human activities from image and video, we do not discuss these works here.

Smith et al. [124] use a number of specifically selected features. Some of these are low-level and deal with color and movement. Others are high-level and are obtained from head and hand regions. A boosting scheme is used that takes into account the history of the activity performance. The work by Vitaladevuni et al. [141] is inspired by the observation that human activities differ in accelerating and decelerating force. They identify reach, yank and throw types. Temporal segmentation into atomic movements described with movement type, spatial location and direction of movement is performed first.

2.1.6 Activity detection

Activity detection approaches do not explicitly model the image representation of a person in the image, nor do they model activity dynamics. Rather, they correlate an observed sequence to labeled video sequences. Such work is aimed at the detection of activities, rather than at their recognition. However, these works share many similarities to those previously discussed and we will describe them briefly in this section.

Zelnik-Manor and Irani describe video segments as bag-of-words encoded over different temporal scales. Each word is the gradient orientation of a local patch. Patches with low temporal variance are ignored, which focuses the representation on moving areas. This restricts the approach to detection against non-moving backgrounds. Ning et al. use Gabor responses instead of gradient orientations. In both works, a histogram distance measure is used.

Shechtman and Irani consider the spatial dimension by correlating space-time patches over different locations in space and time. They use patches that locally describe motion. To avoid calculating the optical flow, a rank-based constraint is used directly on the intensity information of the cuboid. Matikainen et al. approximate this approach but use motion words and a look-up table to allow for faster correlation. More recently, Shechtman and Irani, propose a self-similarity descriptor that correlates local patches. The descriptor is invariant to color and texture and can deal with small spatial variations. A query template is described by an ensemble of all descriptors. Seo et al. use space-time local steering kernels, which can be regarded as a generalization. By applying PCA on a collection of kernels, they obtain the most salient features.

The above methods require that a window is selected through time and space, which makes them computationally expensive. This issue is addressed by Hu et al., who describe a sequence as a collection of windows of different temporal scales and positions and use multiple-instance learning to learn the binary activity classifier. Yuan et al. detect space-time interest points and classify whether each is part of the query activity. An efficient branch-and-bound approach is taken to search for the subvolume that has the maximum of positively labeled points.

Junejo et al. observe that the temporal self-similarity matrix of an activity seen from different viewpoints is very similar. They describe a sequence as a histogram of local descriptors, calculated from the self-similarity matrix. Boiman and Irani take a different approach by describing a sequence as an ensemble of local spatial or spatio-temporal patches.

A similarity score is based on the composition of a query sequence from these patches. Similar sequences require less but larger patches.

Some work assumes motion periodicity, which allows for temporal segmentation by analyzing the self-similarity matrix. Seitz and Dyer introduce a periodicity detection algorithm that is able to cope with small variations in the temporal extent of a motion. They track markers and use an affine distance function. Cluter and Davis perform a frequency transform on the self-similarity matrix of a tracked object. Peaks in the spectrum correspond to the frequency of the motion. The type of activity is determined by analyzing the matrix structure. Polana and Nelson also use Fourier transforms to find the periodicity and temporally segment the video. They match motion features to labeled 2D motion templates.

2.1.7 Summary of Image Representation

Global image representations have proven to yield good results, and they can usually be extracted at low cost. However, their applicability is limited to scenarios where ROIs can be determined reliably. Moreover, they cannot deal with occlusions. Local representations address these issues. Initial work used bag-of-feature representations but more recent work takes into account spatial and temporal correlations between patches. Still, the question of how to deal with more severe occlusions has largely been ignored.

Most of the reported work is restricted to fixed and known viewpoints, which severely limits its applicability. The use of multiple view-dependent activity models solves this issue but at the cost of increased training complexity. Recently, researchers have begun to address the recognition of activities from viewpoints for which there is no corresponding training data.

Regarding classification, we discussed direct classification and temporal stat-space models. In the former, temporal variations are not explicitly modelled, which proved to be a reasonable approach in many cases. For more complex motions, it is questionable whether this approach is suitable. Generative state-space models such as HMMs can model temporal variations but have difficulty distinguishing between related activities (e.g. jogging and walking). In this respect, discriminative graphical approaches are more suitable. In future work, the flexibility of the classifier with respect to adding or removing activity classes from the repertoire will play a more important role.

Many approaches assume that the video has been readily segmented into sequences that contain one instance of a known set of activity labels. The problem can be depicted by Fig. 2.9. Often, it is also assumed that the location and approximate scale of the person in the video is known or can easily be estimated. The activity detection task is thus ignored, which limits the applicability to situations where segmentation in space and time is possible. While several works have addressed this topic, it remains a challenge to perform activity detection for online applications.

Another aspect of human activity recognition is the current evaluation practice. Publicly available datasets have shaped the domain by allowing for objective comparison between approaches on common training and test data. They also allow for better understanding of methods since researchers are aware of the challenge of each set. However, algorithms may be biased to a particular dataset. This may lead to complex approaches that perform better on a specific dataset but may be less generally applicable.

Also, given the increasing level of sophistication of activity recognition algorithms, larger and more complex datasets should direct research efforts to realistic settings. Initially, datasets were not focused on an application domain. However, activity recognition in surveillance, human-computer interactivity and video retrieval poses different challenges. Human-computer interactivity applications require real-time processing, missed detections in surveillance are unacceptable and video retrieval applications often cannot benefit from a controlled setting and require a query interface. Currently, there is a shift towards a diversification in datasets. Such a diversification is beneficial as it allows for realistic recording settings while focusing on relevant activity classes. Moreover, the use of application-specific datasets allows for the use of evaluation metrics that go beyond precision and recall, such as speed of processing or detection accuracy. Still, the compilation or recording of datasets that contain sufficient variation in movements, recording setting and environmental settings remains challenging and should continue to be a topic of discussion.

Related is the issue of labeling data. For increasingly large and complex datasets, manual labeling will become prohibitive. Automatic labeling using video subtitles and movie scripts is possible in some domains, but still requires manual verification. When using an incremental approach to image harvesting, the initial set will largely affect the final variety of activity performances.

We discuss vision-based human activity recognition in this survey but a multi-modal approach could improve recognition in some domains, for example in movie analysis. Also, context such as background, camera motion, interactivity between persons and person identity provides informative cues.

Given the current state of the art and motivated by the broad range of applications that can benefit from robust human activity recognition, it is expected that many of these challenges will be in the near future. This would be a big step towards the fulfillment of the longstanding promise to achieve robust automatic recognition and interpretation of human activity.

2.2 Machine Learning Review

2.2.1 Direct classification

When an image representation is available for an observed frame or sequence, human activity recognition becomes a classification problem. An activity label or distribution over labels is given for each frame or sequence.

The approaches that we describe in this section do not pay special attention to the temporal domain. They summarize all frames of an observed sequence into a single representation or perform activity recognition for each frame individually. While both of these approaches can deal with variations in executing and recording rate, the temporal order is neglected. Later we will discuss nearest neighbour classification where an observed sequence is compared to labeled sequences or activity class prototypes. A second class of approach is that of the discriminative classifiers. These learn a function that discriminates between two or more classes by directly operating on the image representation. Dimensionality reduction is a common step before the actual classification and is discussed first.

2.2.2 Dimensionality reduction

In many cases, image representations are high-dimensional. This makes matching computationally more expensive. Also, the representation might contain noisy features. It is expected that a more compact, robust feature representation is obtained by embedding the space of image representations onto a lower dimensional space. This embedding can be learned from training data.

PCA is a common linear dimensionality reduction method that has been used by Masoud [83] and Papanikolopoulos [117]. Often, the mapping between full and lower dimensional image representation is better described as a non-linear function. Chin et al. [18] learn manifolds using local linear embedding (LLE). They experiment with different projection functions. Wang and Suter [146] use locality preserving projections (LPP). Isomap is used by Blackburn and Ribeiro [9].

The above dimensionality reduction methods learn the embedding in an unsupervised manner, which does not guarantee good discrimination between classes. Poppe and Poel [104] address this issue and learn discriminative feature transforms between pairs of classes. Jia and Yeung [53] use an embedding that is discriminative both in a spatial and temporal sense. They propose local spatio-temporal discriminant embedding (LSTDE), which maps silouettes of the same class close in the manifold and model temporal relations in subspaces of the manifold.

2.2.3 Nearest neighbor classification

k-Nearest neighbor (NN) classifiers use the distance between the image representation of an observed sequence and those in a training set. The most common label among the *k* closest training sequences is chosen as the classification. For a large training set, such comparisons can be computationally expensive. Alternatively, for each class, an activity prototype can be calculated by taking the mean of all corresponding sequences. The ability to cope with variations in spatial and temporal performance, viewpoint and image appearance depends on the training set, the type of image representation and the distance metric.

NN classification can be either performed at the frame level, or for whole sequences. In the latter case, issues with different frame lengths need to be resolved, for example by using majority voting over all frames in a sequence. 1-NN with Euclidean distance are used by Blank et al. [10] for global features and Batra et al. [6] for histograms of codewords. Euclidean distance might not be the most suitable choice given the type of image representation. Bobick and Davis [12] use Hu moments of different orders of magnitude. Mahalanobis distance is used to take into account the variance of each dimension. Rodriguez et al. [115] describe a method to generate spatio-temporal templates that effectively capture the intra-class variance into a single prototype.

Several authors have used NN classification in combination with dimensionality reduction. Wang and Suter [145] either use the minimum mean frame-wise distance in an embedded space, or a frame-order preserving variant. Turage et al. [136] focus on parametric and non-parametric manifold density functions and describe distance functions for Grassmann and Stiefel manifold embedding. Tran et al. [132] and Poppe and Poel [104] use a learned discriminative distance metric in the NN classification.

It has been observed that many activities can be represented by key poses or prototypes. Sullivan and Carlsson [36] recognize forehand and backhand tennis strokes by matching edge representations to labeled key poses. Wang et al. [149] also use edge representations but learn activity clusters in an unsupervised fashion. They manually provide activity class labels after the clustering. Weinland et al. [156] learn a set of activity key poses as a 3D voxel representations. These methods use only a single frame for activity classification. As many poses are only weakly informative for the activity class, considering a sequence of poses over time is likely to reduce ambiguities. Weinland and Boyer [153] use the minimum distance of each key pose for the frames in the sequences. The set of key poses is discriminatively selected. Lin et al. [73] store prototypes in a tree to allow for efficient matching.

2.2.4 Discriminative classifiers

Discriminative classifiers focus on separating two or more classes, rather than modeling them. Support vector machine (SVM) learn a hyperplane in feature space that is described by a weighted combination of support vectors. SVMs have been used in combination with local representations of fixed lengths, such as histograms of codewords in [70]. Relevance vector machines (RVM) can be regarded as the probabilistic variant of the SVM. Training an RVM usually results in a sparser set of support vectors. They have been used for activity recognition by Oikonomopoulos et al. [97].

In a boosting framework, a final strong classifier is formed by a set of weak classifiers, each of which usually uses only a single dimension of the image representation. Boosting is used in many works, either as a discriminative feature selection step or as the actual classifier. AdaBoost has been used by [95]. LPBoost yields sparser coefficients and is reported to converge faster, and is used in [92]. Smith et al. [124] introduce a variant that uses history information in the boosting scheme.

2.2.5 Temporal state-space models

State-space models consist of states connected by edges. These edges model probabilities between states, and between states and observations. In the models that we discuss in this section, each state summarizes the activity performance at a certain moment in time. An observation corresponds to the image representation at a given time. Temporal state-space models are either generative or discriminative. While they share many characteristics, they are conceptually different. Generative models learn a joint distribution over both observations and activity labels. They thus learn to model a certain class, with all its variations. In contrast, discriminative models learn probabilities of the activity classes conditioned on the observations. They do not model a class but rather focus on differences between classes. We discuss generative and discriminative models respectively. Dynamic Time Warping (DTW) can be regarded as a generative model, but it is used between pairs of sequences. Due to this rather different use, we discuss DTW separately later.

2.2.6 Dynamic time warping

Dynamic time warping is a distance measure between two sequences, possibly with different lengths. It simultaneously takes into account a pair-wise distance between corresponding frames and the sequence alignment cost. For a low alignment cost, two sequences need to be segmented similarly in time and performed at similar rates. Dynamic programming is used to calculate the optimal alignment. Veeraraghavan et al. use DTW for sequences of normalized shape features. As these lie on a spherical manifold, the distance function between shapes is adapted. They also address the alignment by considering the space of temporal warping functions for a given activity. Yao et al. introduce dynamic space-time warping where, in addition to the temporal dimension, sequences are also aligned on image position and scale. A related distance is longest common subsequence (LCS). It only takes into account similar elements of both sequences and results in an increased distance when more inserts or deletions are necessary.

2.3 Joint Activity Segmentation and Classification

Activities are often assumed to be readily segmented in time sequences. This assumption moves the burden of the segmentation from the recognition task, but requires a separate segmentation procedure to have been completed previously. This might not always be realistic. Recent work on action detection addresses this issue.

Also, there can be substantial variation in the rate of performance of an activity. The rate at which the activity records has an important effect on the temporal extent of an activity, especially when motion features are used. A robust human activity recognition algorithm should be invariant to different rates of execution.

Actually, in activity sequence classification, we should first extract features from images or video and decide action on labels based on these features. The process can be depicted by Fig. 2.8.

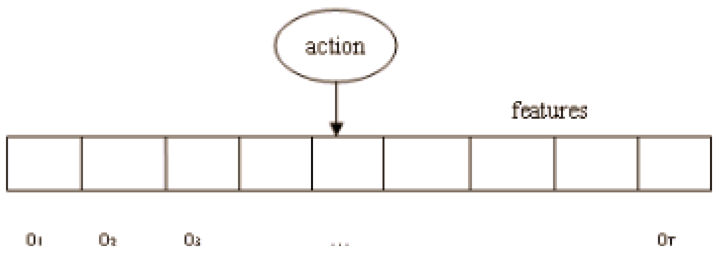


Fig. 2.8 Action sequence with features.

The approaches can be divided into two categories: 1) non-sequential classifiers (direct classifiers) and 2) sequential classifiers. Direct classifiers do not pay special attention to the temporal domain. They summarise all frames of an observed sequence into a single representation or perform activity recognition for each frame individually. Although both of these approaches can deal with variations, the temporal order is neglected. In direct classification, dimensionality reduction is a common step before actual classification. In many cases, image representations are high-dimensional which makes matching computationally more expensive. Also, the representation might contain noisy features. It is expected that a more compact, robust feature representation is obtained by embedding the space of image representation onto a lower dimensional space. This embedding can be learned from training data. Commonly used dimensionality reduction methods are principal component analysis (PCA), local linear embedding (LLE), locality preserving projections (LPP), Isomap [102]. These methods are all unsupervised which does not guarantee good discrimination between classes. Jia and Yeung proposed local spatio-temporal discriminant embedding (LSTDE) which mapped silhouettes of the same class close in the manifold and model temporal relations in subspaces of the manifold [53]. K-nearest neighbour (KNN) classification is usually combined with dimensionality reduction. Wang and Suter either used the minimum mean frame-order preserving variant [147]. Turaga et al. focused on parametric manifold density functions and described distance functions for Grassmann and Stiefel manifold embedding [105]. Tran et al. and Poppe and Poel used a learned discriminative distance metric in the KNN classification [132, 102]. Another useful method is called K-means which is an unsupervised clustering method. K-means clustering of descriptors in the training set gives a vocabulary of primitive event, $h_i, i = 1, \ldots, N$. The number of features with labels h_i in a particular sequence define a feature histogram $H = (h_i, \ldots, h_N)$. This histogram can be used as an alternative representation when recognizing actions in sequences. The feature histograms can then be sent to SVM or other classifiers for classification. Choosing a proper kernel function and parameters, different actions will finally be separated. Sequential classifiers include the famous hidden Markov model (HMM), hidden conditional random field (HCRF), latent structural SVM. We will discuss these methods in a later section.

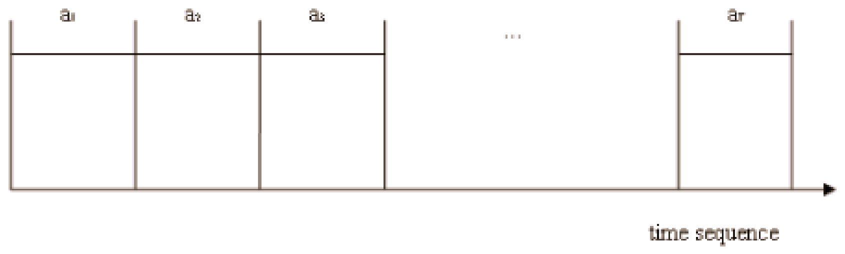


Fig. 2.9 Time sequence which has been segmented and assigned with labels $a_1, \ldots, a_T$.

The baseline approach of this problem is using window slice with N frames in the time sequence such as in a video (N is the size of the window). We assign the stride as M frames (M and N can be the same or different). There is one classifier for each window and we decide which label it belongs to. The approach can be depicted by Fig. 2.10.

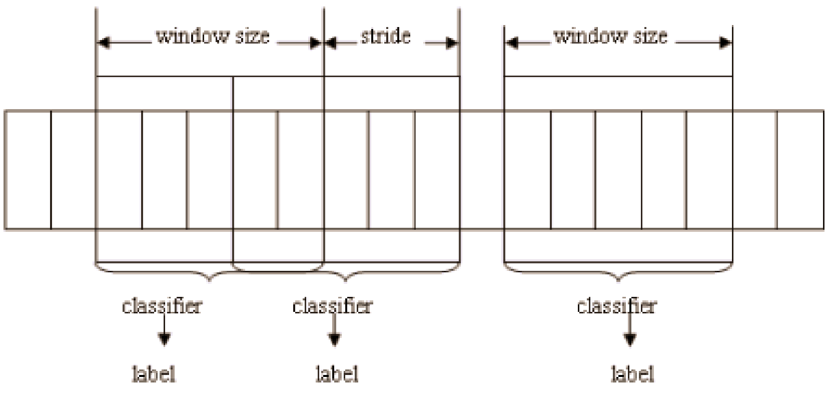


Fig. 2.10 Baseline approach of segmentation and classification jointly.

Our approach for segmentation and classification jointly is similar to HMM, depicted by Fig. 2.11. The main difference is the link between successive activities.

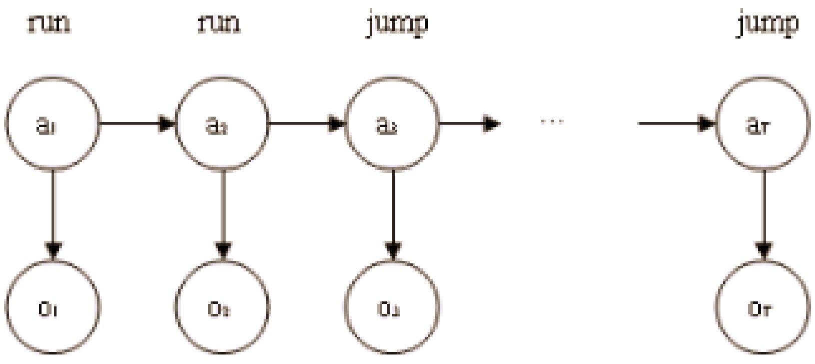


Fig. 2.11 Our approach for segmentation and classification jointly.

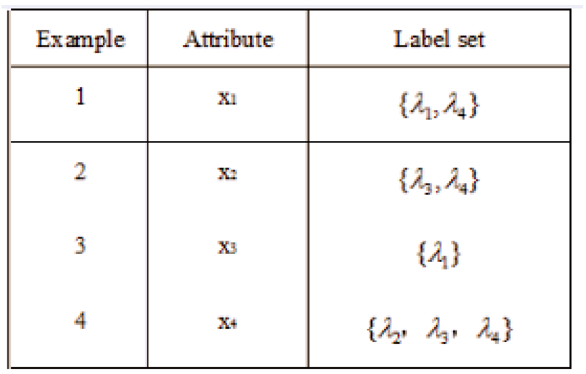
We want to find the probability of $a_1, \ldots, a_T$ based on the measurements $o_1, \ldots, o_T$, which can be formulated as $p(a_{1:T}|o_{1:T}) \propto p(o_{1:T}|a_{1:T}) \cdot p(a_{1:T})$ using Bayesian theory. The most probable labels are $a^*_{1:T} = \text{argmax}_{a_{1:T}} p(a_{1:T}|o_{1:T}) = \text{argmax}_{a_{1:T}} p(o_{1:T}|a_{1:T}) \cdot p(a_{1:T})$. And it is easy to see that $p(o_{1:T}|a_{1:T}) \cdot p(a_{1:T}) = p(a_1) \cdot \prod_{t=1}^{T} p(o_t)|a_t \prod_{t=2}^{T} p(a_t|a_{t-1})$.

2.4 Multi-label Classification

Multi-label Classification (MLC) deals with learning a model that outputs a set of labels out of a set of disjoint labels based on a given instance. We can group the existing methods for multi-label classification into two main categories: 1) problem transformation methods, and 2) algorithm adaptation methods. The first group of methods is algorithm independent. They transform the multi-label classification problem into one or more single-label classification problems. The second group of methods extends specific learning algorithms in order to handle multi-label problems directly.

2.4.1 Problem Transformation

We give an example for problem transformation methods through the multi-label dataset of Fig. 2.12. It consists of four examples which are annotated with one or more labels out of four labels: $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

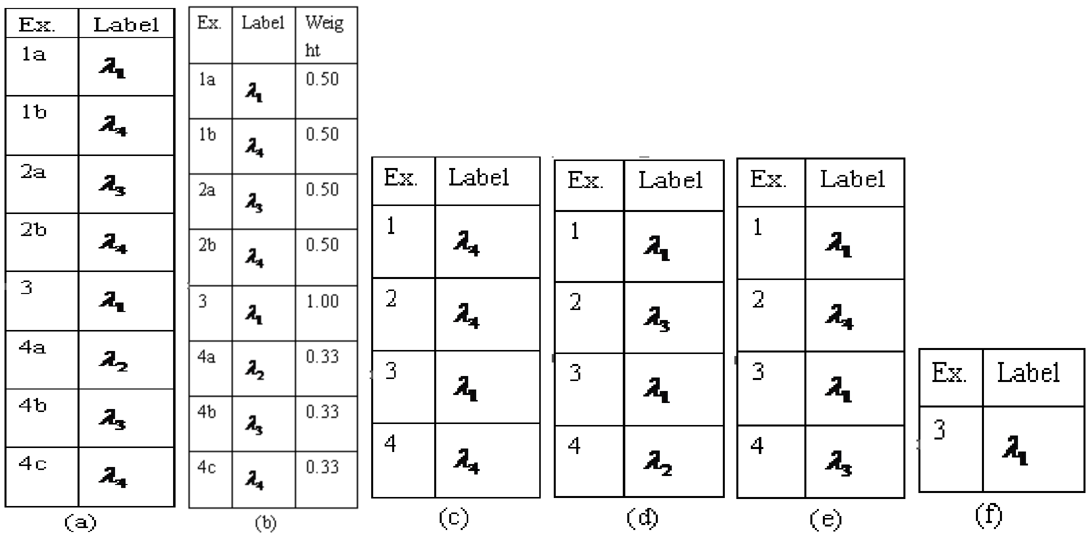


Example	Attribute	Label set
1	x_1	$\{\lambda_1, \lambda_4\}$
2	x_2	$\{\lambda_3, \lambda_4\}$
3	x_3	$\{\lambda_1\}$
4	x_4	$\{\lambda_2, \lambda_3, \lambda_4\}$

Fig. 2.12 Example of a multi-label dataset.

There exist several simple transformations that can be used to convert a multi-label dataset to a single-label dataset with the same set of labels. The trivial approach is to expand the problem from multi-label over q classes to simple-label over 2^q classes, where q is the total number of classes in the training set. However, this is not feasible when q is significant. A single-label classifier that outputs probability distributions over all classes can then be used to learn a ranking. The class with the highest probability will be ranked the first; the class with the second best probability will be ranked the second, and so on.

The *copy* transformation replaces each multi-label example (x_i, Y_i) with $|Y_i|$ examples (x_i, λ_j), for every $\lambda_j \in |Y_i|, j = 1, \ldots, q$. A variation of this transformation, called *copy-weight*, associates a weight of $\frac{1}{Y_i}$ to each of the produced examples. The *select* methods replace



Ex.	Label
1a	λ_1
1b	λ_4
2a	λ_3
2b	λ_4
3	λ_1
4a	λ_2
4b	λ_3
4c	λ_4

(a)

Ex.	Label	Weig ht
1a	λ_1	0.50
1b	λ_4	0.50
2a	λ_3	0.50
2b	λ_4	0.50
3	λ_1	1.00
4a	λ_2	0.33
4b	λ_3	0.33
4c	λ_4	0.33

(b)

Ex.	Label
1	λ_4
2	λ_4
3	λ_1
4	λ_4

(c)

Ex.	Label
1	λ_1
2	λ_3
3	λ_1
4	λ_2

(d)

Ex.	Label
1	λ_1
2	λ_4
3	λ_1
4	λ_3

(e)

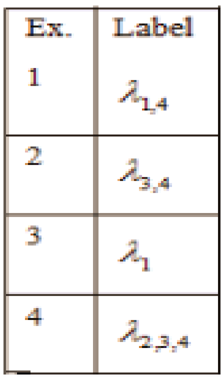
Ex.	Label
3	λ_1

(f)

Fig. 2.13 Transformation of the dataset using (a) *copy*, (b) *copy-weight*, (c) *select-max*, (d) *select-min*, (e) *select-random* (one of the possible) and (f) *ignore*.

$|Y_i|$ with one of its members. The label could be the most (*select-max*) or least (*select-min*) frequent among all examples. It could also be randomly selected (*select-random*). Finally, the *ignore* transformation simply discards every multi-label example and just keeps the non multi-labels. Fig. 2.13 shows the transformed datasets using these simple transformations.

Label powerset (LP) is a simple but effective problem transformation method that works as follows: It considers each unique set of labels that exist in a multi-label training set as one class of a new single-label classification task. Fig. 2.14 shows the result of transforming the data of Fig. 2.12 using LP.



Ex.	Label
1	$\lambda_{1,4}$
2	$\lambda_{3,4}$
3	λ_1
4	$\lambda_{2,3,4}$

Fig. 2.14 Transformed data using label powerset method.

Given a new instance, the single-label classifier of LP outputs the most probable class, which is actually a set of labels.

Ex.	Label
1	λ_1
2	$\neg\lambda_1$
3	λ_1
4	$\neg\lambda_1$

(a)

Ex.	Label
1	$\neg\lambda_2$
2	$\neg\lambda_2$
3	$\neg\lambda_2$
4	λ_2

(b)

Ex.	Label
1	$\neg\lambda_3$
2	λ_3
3	$\neg\lambda_3$
4	λ_3

(c)

Ex.	Label
1	λ_4
2	λ_4
3	$\neg\lambda_4$
4	λ_4

(d)

Fig. 2.15 Dataset produced by the BR method.

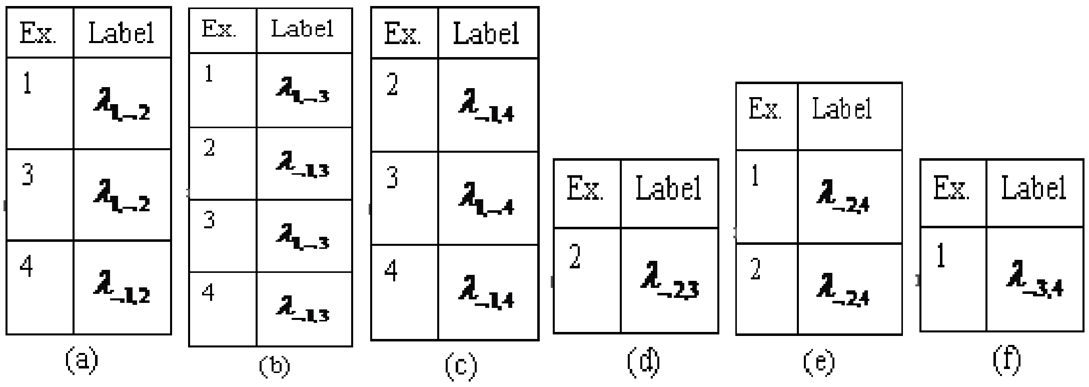
The computational complexity of LP with respect to q depends on the complexity of the basic classifier with respect to the number of classes, which equals to the number of distinct label sets in the training set. This number is upper bounded by $\min(m, 2^q)$, where m is the total number of instances. Although it is much smaller, it still faces an important complexity problem, especially for large value of m and q. The large number of classes, many of which are associated with very few examples, makes the learning process difficult as well.

The pruned problem transformation (PPT) method extends LP to deal with the aforementioned problems [114]. It prunes away label sets that occur less times than a small user-defined threshold (e.g. 2 or 3) and optionally replaces their information by introducing disjoint subsets that exist more times than the threshold.

The random K-labelsets (RAKEL) method constructs an ensemble of LP classifiers [135]. Each LP classifier is trained using a different small random subset of the set of labels. RAKEL manages to take labels correlations into account by this method and in the meantime it can avoid LP's problems. A ranking of the labels is produced by averaging the zero-one predictions of each model.

Binary relevance (BR) is a popular problem transformation method that learns q binary classifiers, one for each different label in L. It transforms the original dataset into q datasets $D_{\lambda_j}, j = 1, \ldots, q$ that contain all the examples of the original dataset, labeled positive if the label set of the original example contains λ_j and negative otherwise. For the classification of a new instance, BR outputs the union of the labels λ_j that are positively predicted by the q classifiers. Fig. 2.15 shows the four datasets that are constructed by BR when applied to the dataset of Fig. 2.12. The limitation of the BR method is that it predicts labels independently rather than jointly.

Ranking by pairwise comparison (RPC) transforms the multi-label dataset into $q(q-1)/2$ binary label datasets [47], one of each pair of labels $(\lambda_i, \lambda_j), 1 \leq i < j \leq q$. Each dataset contains those examples of D that are annotated by at least one of the two corresponding



(a)

Ex.	Label
1	$\lambda_{1,\neg 2}$
3	$\lambda_{1,\neg 2}$
4	$\lambda_{\neg 1,2}$

(b)

Ex.	Label
1	$\lambda_{1,\neg 3}$
2	$\lambda_{\neg 1,3}$
3	$\lambda_{1,\neg 3}$
4	$\lambda_{\neg 1,3}$

(c)

Ex.	Label
2	$\lambda_{\neg 1,4}$
3	$\lambda_{1,\neg 4}$
4	$\lambda_{\neg 1,4}$

(d)

Ex.	Label
2	$\lambda_{\neg 2,3}$

(e)

Ex.	Label
1	$\lambda_{\neg 2,4}$
2	$\lambda_{\neg 2,4}$

(f)

Ex.	Label
1	$\lambda_{\neg 3,4}$

Fig. 2.16 Dataset produced by the RPC method.

labels, but not both. A binary classifier that learns to discriminate between the two labels is trained from each of these datasets. Given a new instance, all binary classifiers are invoked, and a ranking is obtained by counting the votes received by each label. Fig. 2.16 shows the datasets that are constructed by RPC when applied to the dataset of Fig. 2.12. The multi-label pairwise perceptron (MLPP) algorithm is an instantiation of RPC using perceptrons for the binary classification tasks [78].

Calibrated label ranking (CLR) extends RPC by introducing an additional virtual label, which acts as a natural breaking point of the ranking into relevant and irrelevant sets of labels [38]. CLR manages to solve the complete MLR task. The binary models that learn to discriminate between the virtual label and each of the other labels correspond to the models of BR. Each of the examples is annotated with a given label that is considered as positive for this label and negative for the virtual label, while each example is not annotated with a label that is considered for it and positive for the virtual label. When applied to the data of Fig. 2.12, CLR would construct both the datasets of Fig. 2.16 and Fig. 2.15.

The INSDIF algorithm computes a prototype vector for each label, by averaging all instances of the training set that belongs to this label [169]. After that, every instance is transformed to a bag of q instances, each equals to the difference between the initial instance and one of the prototypes to learn from the transformed dataset.

In the following work, we adopt the binary relevance (BR) method for its appealing simplicity.

2.5 Sequential Labeling

The previous discussion on multi-label classification mainly refers to classes whose data are assumed to be independent and identically distributed (i.i.d.). For many applications, however, the i.i.d. assumption will be a poor one. Here we consider a particularly important class of such datasets, namely sequential data. These often arise through measurement of time series, for example the rainfall measurements on successive days at a particular location, or the daily values of a currency exchange rate, or the acoustic feature at successive time frames used for speech recognition [8].

Given a data sequence O, we want to assign it to the best class c^* out of C classes. Like in any other classification problems, one can learn a discriminant function $c^* = f(O)$, or a posterior probability $p(c|O)$ and find c^* as $\text{argmax}_c\, p(c|O)$. Alternatively, we can use a generative approach and Bayes' inversion rule $c^* = \text{argmax}_c(p(O|c) \cdot p(c))$.

2.5.1 Discrete Markov Models

The easiest way to treat sequential data would be simply to ignore the sequential aspects and treat the observations as i.i.d.. Such an approach, however, would fail to exploit the sequential patterns in the data, such as correlations between observations that are close in the sequence. Suppose, for instance, that we observe a binary variable denoting whether on a particular day it rained or not. Given a time series of recent observations of this variable, we wish to predict whether it will rain the next day. If we treat the data as i.i.d., then the only information we can collect from the data is the relative frequency of rainy days. However, in practice we know that the weather often stays the same for several days. Observing whether or not it rains today is very helpful in predicting if it will rain tomorrow.

A model to describe the evolution of a system can be described as follows: 1) at any time frame t, the system is in a state q_t, which can be in one of N possible values, $S = \{s_1, s_2, \ldots, s_N\}$. 2) at regularly spaced times, the system undergoes a transition to a new state q_{t+1}. 3) transition between states can be described probabilistically. By discrete we mean both time-discrete and with a finite number of states. For the special case of a discrete, first-order Markov chain, this probabilistic description can be formulated as:

$$p(q_t = s_j | q_{t-1} = s_i, q_{t-2} = s_k, \ldots) = p(q_t = s_j | q_{t-1} = s_i) \tag{2.1}$$

Joint probability of the state over T frames is $p(Q) = p(q_1, \ldots, q_t, \ldots, q_T)$. The general case (chain rule) is

$$\begin{aligned}
&p(q_1,\ldots,q_t,\ldots,q_T)\\
&= p(q_T|q_1,\ldots,q_{T-1})\cdot p(q_1,\ldots,q_{T-1})\\
&= p(q_T|q_1,\ldots,q_{T-1})\cdot p(q_{T-1}|q_1,\ldots,q_{T-2})\cdot p(q_1,\ldots,q_{T-2})\\
&= \ldots
\end{aligned} \tag{2.2}$$

With the Markov hypothesis, we have:

$$\begin{aligned}
&p(q_1,\ldots,q_t,\ldots,q_T)\\
&= p(q_T|q_{T-1})\cdot p(q_1,\ldots,q_{T-1})\\
&= p(q_T|q_{T-1})\cdot p(q_{T-1}|q_{T-2})\cdot p(q_1,\ldots,q_{T-2})\\
&= \ldots\\
&= p(q_1)\cdot\prod_{t-2}^{T} p(q_t|q_{t-1})
\end{aligned} \tag{2.3}$$

2.5.2 Hidden Markov Models

So far we consider that each state for the Markov model can be observed already. In many case of interest, the state of a system cannot be directly observed but rather inferred through measurements of other variables or observation as a probabilistic function of the state. This implies that the state of a system at any given time can be treated as a hidden random variable while a set of measurements which we can acquire from the system are treated as observed variables.

A *hidden Markov Model* (HMM) [107] is a probabilistic model for a sequence of observations $O = o_1,\ldots,o_t,\ldots,o_T$ and the corresponding hidden states $Q = q_1,\ldots,q_t,\ldots,q_T$. In HMM, the probability of the observation is obtained as marginal of joint probability of observations and states $p(O) = \sum_Q p(O,Q)$. The states are discrete and symbolic in value: each state q_t takes value in a finite set of N symbols, $S = s_1, s_2, \ldots, s_N$, sometimes for simplicity we just note $S = 1,2,\ldots,N$, yet with exactly the same symbolic meaning. For the moment, we assume that the observations are also discrete, K is the number of distinct observation symbols per state. The observation symbols correspond to the output of the system being modeled. We denote the individual symbols as $V = v_1,\ldots,v_K$.

Given appropriate values of N, K, A, B and π, the HMM can be regarded as a generator to give an observation sequence $O = o_1, o_2, \ldots, o_T$ (where each observation o_t is one of the symbols from V, and T is the number of observations in the sequence) as follows:

1) Choose an initial state $q_1 = S_i$, according to the initial state distribution π.

2) Set $t = 1$.

3) Choose $o_t = v_k$ according to the symbol probability distribution in state S_i, i.e., $b_i(k)$.

4) Transit to a new state $q_{t+1} = s_j$ according to the state transition probability distribution for state S_i, i.e. a_{ij}.

5) Set $t = t + 1$; return to step 3 if $t < T$; otherwise terminate the procedure.

The above procedure can be used as both a generator of measurements and as a model for how a given measurements sequence is generated by an appropriate HMM.

We can see that a complete HMM requires specification of two model parameters (N and K), specification of observation symbols, and specification of the three probability measures A, B and π.

Modeling $P(O)$ or $p(O,Q)$ directly can be challenging because of the high dimensionality of such joint probabilities. On the other hand, assuming that $p(O) = p(o_1) \cdot \ldots \cdot p(o_T)$ would dismiss the sequential nature of these data.

There are two fundamental hypotheses in an HMM: 1) Markov state transition. More precisely, q_t given q_{t-1} is independent of the other previous variables, $p(q_t)|q_{t-1}, o_{t-1}, \ldots, q_1, o_1 = p(q_t|q_{t-1})$. 2) independent of each observation given its state, $p(o_t|q_T, o_T, \ldots, q_{t+1}, o_{t+1}, q_t, o_t, q_{t-1}, o_{t-1}, \ldots$ $p(o_t|q_t)$.

An HMM can be depicted rolled-out in time as a graphical model, the absence of links means conditional independence.

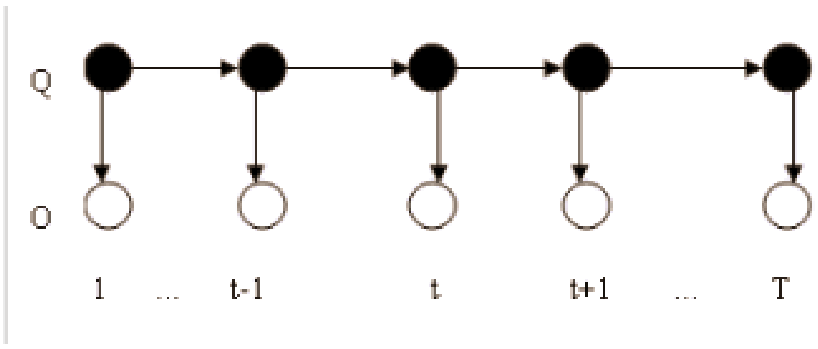


Fig. 2.17 Hidden Markov Model

The generative model is described as Marginal and Joint models and defined as follows:

$$p(O) = \sum_Q p(O,Q) \tag{2.4}$$

$$\begin{aligned} p(O,Q) &= p(o_1,\dots,o_t,\dots,o_T,q_1,\dots,q_t,\dots,q_T) \\ &= p(o_T|q_T)\cdot p(q_T|q_{T-1})\dots p(o_t|q_t)\cdot p(q_t|q_{t-1})\dots \\ &= p(q_1)\cdot(\prod_{t=2}^{T} p(q_t|q_{t-1}))\cdot(\prod_{t-1}^{T} p(o_t|q_t)) \end{aligned} \tag{2.5}$$

The HMM parameters are collectively noted as λ, where $\lambda = A, B, \pi$. A is the state transition probability which is an $N * N$ matrix $a_{ij} = p(q_t = s_j|q_{t-1} = s_i)$, $1 \le i, j \le N$. B is the observation probability which is an $N * K$ matrix $b_j(k) = p(o_t = v_k|q_t = s_j)$, $1 \le j \le N$, $1 \le k \le K$. π is the initial state probability which is $N * 1$ vector $\pi = p(q_1 = s_i)$, $1 \le i \le N$.

Three Canonical Problems of HMM

Given the form of HMM in the previous section, there are three canonical problems of interest about HMM that must be solved for the model before it can be used in the real-world applications, each of which have well-worked solutions:

1) Given the observation sequence $O = o_1, o_2, \dots, o_T$ and the model$\lambda = (A, B, \pi)$, how do we efficiently compute likelihood $p(O|\lambda)$ (the probability of the observation sequence given the model), which is also called evaluation. O is basically our samples, and $p(O|\lambda)$ is its probability in the model. Being able to compute $p(O|\lambda)$ allows us to perform Maximum Likelihood (ML) or Maximum A Posteriori (MAP) classification.

2) Given the observation sequence $O = o_1, o_2, \dots, o_T$ and the model $\lambda = (A, B, \pi)$, how do we find the best corresponding state sequence $Q = q_1, q_2, \dots, q_T$ which is also known as decoding or inference. In certain scenarios, states may have a physical interpretation (i.e. the digit of a phone number), observations are treated as noisy measurements of the states, and state decoding is equivalent to filtering out the noise.

3) Given $O = o_1, o_2, \dots, o_T$, find $\lambda = (A, B, \pi)$ maximizing $p(O|\lambda)$ which is also called likelihood estimation or learning/training. This is equivalent to density estimation. Typically, we need to learn a model from a set of samples before we can use it for 1) and 2).

For 1), this is the evaluation problem, namely given a model and a sequence of observations, how to compute the probability that the observed sequence is produced by the model. This problem can also be regarded as scoring how well a given model matches a given observation sequence, which is much more meaningful. For example, if we consider the case in which we are trying to choose among several competing models, the solution to this problem allows us to choose the model which best matches the observation sequence.

For 2), this is the one in which we try to uncover the hidden part of the model, i.e. to find the "correct" state sequence. It should be clear that only the "optimal" state sequence could be found for practical situations. We usually use an optimality criterion to solve this problem. As we will see, there are several reasonable optimality criteria that can be imposed, and hence the choice of criterion is a strong function of the intended use for the uncovered state sequence. Typical uses might be to learn about the structure of the model, to find the optimal state sequence for sequential labeling , or continuous speech recognition, etc.

For 3), this is the one in which we try to optimise the model parameters in order to best describe how a given observation sequence comes out. The observation sequence used to adjust the model parameters is called a learning/training sequence since it is used to train the HMM. The training problem is the crucial one for most applications of HMM, since it allows us to optimally adapt model parameters to observed training data, i.e. to create the best models for real-world problems.

Let's consider the following simple word speech recognition problem. For each word of a W word vocabulary, we want to design a separate N-state HMM. We represent the speech signal of a given word as a time sequence of coded spectral vectors. We assume that the coding is done using a spectral codebook with K unique spectral vectors, hence each observation is the index of the spectral vector closest to the original speech signal. Therefore, for each vocabulary word, we have a training sequence consisting of a number of repetitions of sequences of codebook indices of the word. The first task is to build individual word models. This task is done by using the solution to Problem 3 to optimally estimate model parameters for each word model. To develop an understanding of the physical meaning of the model states, we use the solution to Problem 2 to segment each of the word training sequences into states, and then study the properties of the spectral vectors that lead to the observations occurring in each state. The goal here is to make refinements on the model (e.g. more states, different codebook size, etc.) in order to improve its capability of modeling the spoken word sequences. Finally, once the set of W HMMs have been designed, optimized and thoroughly studied, recognition of an unknown word is performed using the solution to Problem 1 to score each word model based on the given test observation sequence, and selecting the word whose model score is the highest (i.e. the highest likelihood).

Solution to problem 1: We want to calculate the probability of the observation sequence $O = o_1, o_2, \ldots, o_T$, given the model π, i.e. $p(O|\lambda)$. The most straightforward way of doing this is through enumerating every possible state sequence of length T (the number of observation). Consider one such fixed state sequence $Q = q_1, q_2, \ldots, q_T$, where q_1 is the initial state. The probability of the observation sequence O for the state sequence is

$$p(O|Q,\lambda) = \sum_{t=1}^{T} p(o_t|q_t,\lambda) \tag{2.6}$$

where we have assumed statistical independence of observations. Thus we get

$$p(O|Q,\lambda) = b_{q_1}(o_1) \cdot b_{q_2}(o_2) \ldots b_{q_T}(o_T) \tag{2.7}$$

The probability of such a state sequence Q can be written as

$$p(Q|\lambda) = \pi_{q_1} \cdot a_{q_1,q_2} \cdot a_{q_2,q_3} \ldots a_{q_{T-1},q_T} \tag{2.8}$$

The joint probability of O and Q, i.e. the probability that O and Q occur simultaneously is simply the product of the above two terms, i.e.

$$p(O,Q|\lambda) = p(O|Q,\lambda) \cdot p(Q,\lambda) \tag{2.9}$$

The probability of O (given the model) is obtained by summing this joint probability over all possible state sequences q giving

$$p(O|\lambda) = \sum_{\forall Q = q_1, q_2, \ldots, q_T} \pi_{q_1} \cdot a_{q_1,q_2} \cdot b_{q_1}(o_1) \cdot a_{q_2,q_3} \cdot b_{q_2}(o_2) \ldots a_{q_{T-1},q_T} \cdot b_{q_T}(o_T) \tag{2.10}$$

The interpretation of the computation in the above equation is the following. Initially at time $t = 1$ we are in state q_t with probability π_{q_t}, and generate the symbol o_t in this state with probability $b_{q_t}(o_1)$. The clock changes from time t to $t+1$ (t=2) and we make a transition to state q_2 from state q_1 with probability $a_{q_1 q_2}$, and generate symbol o_2 with probability $b_{q_2}(o_2)$. This process continues in this manner until we make the list transition at time T from state q_{T-1} to state q_T with probability $a_{q_{T-1} q_T}$ and generate symbol o_T with probability $b_{q_T}(o_T)$.

Unfortunately, its direct definition is in the order of $2T \cdot N^T$ calculations, since at every $t = 1,2,\ldots,T$, there are N possible states which can be reached, i.e. there are N^T different possible assignments for Q. This makes naive evaluation impractical, even for small values of N and T: e.g., for $N = 5$ and $T = 100$, there are in the order of $2 \cdot 100 \cdot 5^{100} = 10^{72}$ computations. Luckily, the complexity can be reduced from exponential to linear in T. The algorithm is known as the *forward procedure* (or *backward procedure*).

The *forward procedure*: Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = p(o_1, o_2, \ldots, o_t, q_t = S_i|\lambda) \tag{2.11}$$

i.e. the probability of the partial observation sequence $o_1, o_2, \ldots, o_t$ (until time t) and state s_i at time t, given the model λ. We can solve for $\alpha_t(i)$ inductively, as follows:

Step 1: Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1),\ 1 \leq i \leq N \tag{2.12}$$

Step 2: Induction:

$$\alpha_{t+1}(j) = \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}),\ 1 \leq t \leq T-1,\ 1 \leq j \leq N \tag{2.13}$$

Step 3: Termination:

$$p(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.14}$$

In a similar manner, we can consider a backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = p(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = S_i, \lambda) \tag{2.15}$$

i.e. the probability of the partial observation sequence from $t+1$ to the end, given state s_i at time t and the model λ. Again we can solve for $\beta_t(i)$ inductively, as follows:

Step 1: Initialization:

$$\beta_T(i) = 1,\ 1 \leq i \leq N \tag{2.16}$$

Step 2: Induction:

$$\beta_t(i) = \sum_{j+1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j),\ t = T-1, T-2, \ldots, 1, 1 \leq i \leq N \tag{2.17}$$

Solution to problem 2: our hypothesis is that q_t conditioned on q_{t-1} is independent of the other previous variables. Yet, this does not apply to the future ones. This means that one needs to use the entire observation sequence O to optimally estimate q_t as $q_t^* = \operatorname{argmax} p(q_t|O)$, estimating q_t based on observations up to O_t only.

Assume now that we are after not just the optimal q_t, but the optimal state sequence $q_1^* \ldots q_T^*$, i.e. $Q^* = \operatorname{argmax}_Q p(Q|O)$: the maximum of the joint Q is not the same as the union of the maxima over each marginal q_t. Note also that $\operatorname{argmax}_Q p(Q,O) = \operatorname{argmax}_Q p(Q|O)$ since $p(O)$ plays no role in the maximization. Normally, $p(Q|O)$ is simpler than $p(Q,O)$ since it doesn't contain a model for $p(O)$. However, in HMM, $p(Q,O)$ is the obvious quantity and $p(Q|O)$ is obtained as $p(Q,O)/p(O)$. The solution is provided by the *Viterbi algorithm*.

Viterbi Algorithm: To find the single best state sequence $Q = q_1, q_2, \dots, q_T$, for the given observation sequence $O = o_1, o_2, \dots, o_T$, we need to define the quantity

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} p(q_1, q_2, \dots, q_t = i, o_1, o_2, \dots, o_t | \lambda) \tag{2.18}$$

i.e. $\delta_t(i)$best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state s_i. By induction we have

$$\delta_{t+1}(j) = \max_j \delta_t(i) a_{ij} \cdot b_j(o_{t+1}) \tag{2.19}$$

To actually retrieve the state sequence, we need to keep track of the argument which maximises Eq. (2.19), for each t and j. We do this via the array $\psi_t(i)$. The complete procedure for finding the best state sequence can now be stated as follows:

Step 1: Initialization:

$$\delta_1(i) = \pi_i b_i(o_1), \ \psi_1(i) = 0, \ 1 \le i \le N \tag{2.20}$$

Step 2: Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} \delta_{t-1}(i) a_{ij} b_j(o_t), \ 2 \le t \le T, \ 1 \le j \le N \tag{2.21}$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} \delta_{t-1}(i) a_{ij}, \ 2 \le t \le T, \ 1 \le j \le N \tag{2.22}$$

Step 3: Termination:

$$q_T^* = \operatorname*{argmax}_{1 \le i \le N} \delta_T(i), \ P^* = \max_{1 \le j \le N} \delta_T(i) \tag{2.23}$$

Step 4: Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \ t = T-1, T-2, \dots, 1 \tag{2.24}$$

Solution to problem 3: So far the most difficult problem of HMM is to determine a method to adjust the model parameters $\lambda = (A, B, \pi)$ to maximise the probability of the observation sequence given the model. There is no known way to analytically solve how the model maximizes the probability of the observation sequence. In fact, given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. However, we can choose $\lambda = (A, B, \pi)$ so that $p(O|\lambda)$ is locally maximised using an iterative procedure such as the Baum-Welch method, or equivalently through the EM (expectation-modification) method, or by using gradient techniques.

Although use of HMM technology has contributed greatly to recent advances in applications such as speech recognition, handwriting recognition and so on, there are some inherent limitations of this type of statistical model. A major limitation is the assumption that successive observations are independent, and therefore the probability of a sequence of observations $p(o_1, o_2, \ldots, o_T)$ can be written as a product of probabilities of individual observations, i.e. $p(o_1, o_2, \ldots, o_T) = \sum_{t=1}^{T} p(o_t)$. Another limitation is the assumption that the distributions of individual observation parameters can be well represented as a mixture of Gaussian or autoregressive densities. Finally the Markov assumption itself, i.e. that the probability of being in a given state at time t only depends on the state at time $t-1$ is clearly inappropriate where dependencies often extend through several states. However, this type of statistical model has been tested very well for certain applications in spite of these limitations.

2.6 Support Vector Machine

2.6.1 Binary SVM

The support vector machine (SVM) was originally proposed by Vapnik and Cortes around 1995 (soft-margin version) in order to deal with binary classification problems [21]. SVM implements the following idea: it maps the input vectors into some high dimensional feature space through some non-linear mapping chosen a priori. One needs to construct a linear decision hyperplane in this feature space with special properties in order to ensure that SVM has a high generalization ability.

The limitation of maximum likelihood estimation and minimum risk estimation is that training samples should be infinite. If given limited samples, the empirical risk minimization (ERM) rule performs badly in classification. Traditional learning introduces generalization bound, including two parts: empirical risk and trust risk. Empirical risk stands for error on given samples and trust risk stands for how much we can trust the classifier on samples which are not given. SVM aims to minimize the sum of empirical risk and trust risk together.

It is normally thought that SVM has two advantages: the first is that SVM needs fewer samples. It doesn't mean the absolute value of a number of samples but SVM requires relatively fewer samples if we consider the complexity of the problem. The second one is its non-linear feature. SVM is good at dealing with non-linear separated data thanks mainly to the use of kernels.

There is one conceptual and one technical problem arising in SVM. The conceptual problem is how to find a separating hyperplane that has a good generalisation ability: the dimensionality of the feature space is large and there is no guarantee that all hyperplanes separating the training data have good generalisation ability. The technical problem is how to compute such high dimensional spaces efficiently, for example, to construct polynomials of degree 4 or 5 in a 200 dimensional space. It may be that we need to construct hyperplanes in a billion dimensional feature space.

The conceptual problem was solved by Vapnik in 1965 for the case of optimal hyperplanes for separable classes. The optimal hyperplane is defined as the linear decision function with maximum margin between the vectors of the two classes. Vapnik has shown that to construct such optimal hyperplanes one does not need to take into account all training data, but instead just a small subset. This set of training data is called *support vectors*, which determine the margin. The reason why we are so interested in the support vector is that if the training vectors are separated correctly with an optimal hyperplane, the expectation value of the probability to commit an error on a test sample is bounded by the ratio between the expected value of the number of support vectors and the number of training vectors.

$$E[pr(error)] \leq \frac{E[numberofsupportvectors]}{numberoftrainingvectors} \tag{2.25}$$

This bound does not explicitly indicate the dimensionality of the separation space. From this bound , we can easily see that if there is an optimal hyperplane which can be constructed from only a subset of the training data (the support vectors), it usually generalises well, even in an infinite dimensional space.

To start with, we assume that the training dataset is linearly separable in feature space, so that by definition there exists at least one choice of the parameters of a vector w and a scalar b such that a function satisfies

$$\begin{aligned} w \cdot x_i + b &\geq 1 \ if \ y_i = 1 \\ w \cdot x_i + b &\leq -1 \ if \ y_i = -1 \end{aligned} \tag{2.26}$$

So the inequalities satisfy all elements of the training set. Below we write the inequalities in the form

$$y_i(w \cdot x_i + b) \geq 1, \ i = 1, 2, \ldots, n \tag{2.27}$$

The optimal hyperplane

$$w_0 \cdot x + b_0 = 0 \tag{2.28}$$

is the unique one which separates the training data with a maximal margin: it determines the direction $w/|w|$ where the distance between the projections of the training vectors of two different classes is maximal.

We can use Fig. 2.18 to indicate this situation.

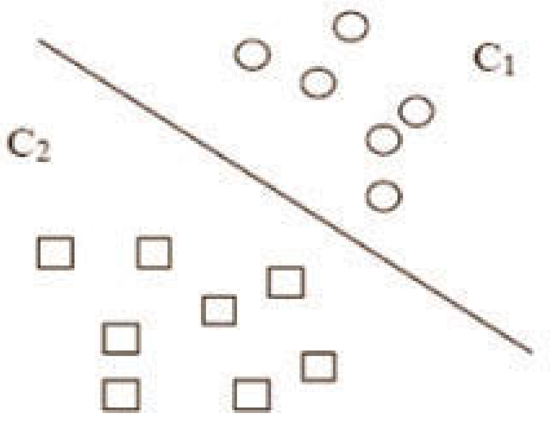


Fig. 2.18 Binary classification problem

There are two classes, indicated by squares and circles separately. The line between two classes is called hyperplane (decision boundary). It is a linear function that classifies different data sets. We can easily see that there exist multiple solutions, and we should try to find the one that will give the smallest generalization error. The support vector machine approaches this problem through the concept of margin, which is defined to be the smallest distance between the decision boundary and any of the samples.

However, although the conceptual problem has been solved and the optimal hyperplane has a good generalization, we still face the problem of the technical problem of how to treat the high dimensional feature space. In 1992, it was shown that one can consider changing the step of constructing a decision function in operation: instead of making a non-linear transformation of the input space vectors followed by support vectors using dot-product in feature space, one can first compare two vectors in input space, for example taking their dot-product, and then making a non-linear transformation of the value of the dot-product (see Fig. 2.19). By doing this, one enables the construction of rich classes of decision functions, for example polynomial decision function of arbitrary degree.

The technique of SVM was originally developed to handle the separable case, which means the restricted training data with non error. However, SVM has been extended for non

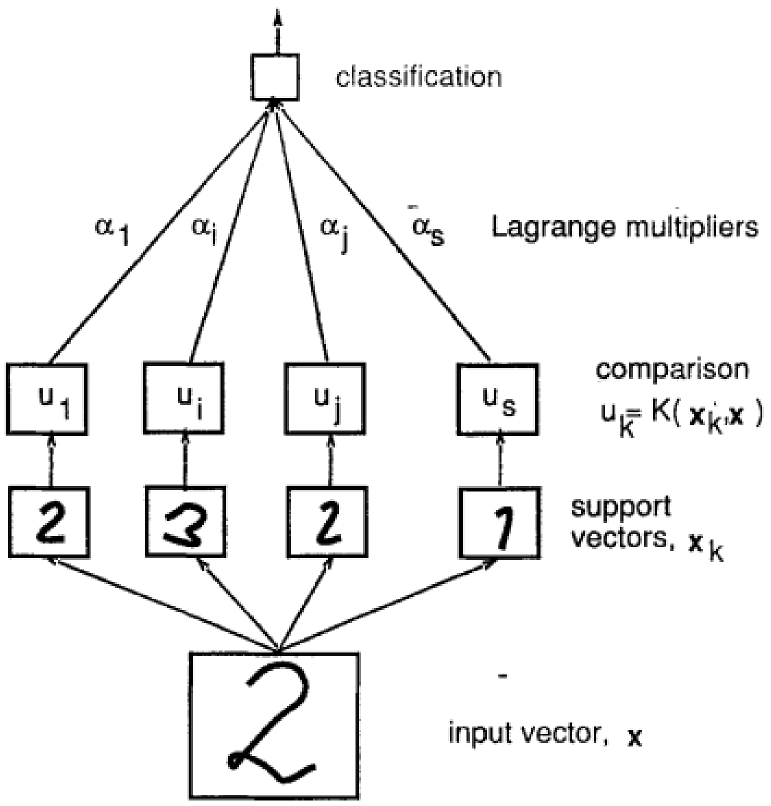


Fig. 2.19 Classification of an unknown pattern by an SVM. The pattern is in input space compared to support vectors. The resulting values are non-linearly transformed. A linear function of these transformed values determines the output of the classifier

separable cases because many real problems cannot be separated without error. Considering this, some researchers treat SVM as a new class of learning machine, as powerful and universal as the neural network.

To construct the optimal hyperplane

$$w_0 \cdot x + b_0 = 0 \tag{2.29}$$

which separates a set of training examples $(y_1, x_1), (y_2, x_2), \ldots, (y_n, x_n)$, we have to minimise a functional $\frac{1}{2}||w||^2$, subject to the constraints

$$y_i(w \cdot x_i + b) \geq 1, \; i = 1, 2, \ldots, n \tag{2.30}$$

In the following, we show how this constrained minimum can be found, mainly following [21]

It has been shown that we can construct a Lagrangian function to use a standard optimisation technique

$$L(w,b,\Lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(w\cdot x_i + b) - 1] \tag{2.31}$$

where $\Lambda^T = (\alpha_1, \alpha_2, \ldots, \alpha_n), \alpha_i \geq 0, i = 1, \ldots, n$ is the vector of Lagrangian multipliers corresponding to the constraints Eq. (2.30).

It is known that the solution to the optimisation problem is determined by the saddle point of this Lagrangian in the $2n+1$ dimensional space of w, Λ and b, where the minimum should be taken with respect to the parameters w and b, and the maximum should be taken with respect to the Lagrangian multipliers Λ.

At the point of the minimum with respect to w and b, we can obtain:

$$\frac{\partial L(w,b,\Lambda)}{\partial w}|_{w=w_0} = w_0 - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \tag{2.32}$$

$$\frac{\partial L(w,b,\Lambda)}{\partial w}|_{b=b_0} = \sum_{\alpha_i} y_i \alpha_i = 0 \tag{2.33}$$

We can also get from equality Eq. (2.32)

$$w_0 = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.34}$$

This equation indicates that the solution to the optimal hyperplane can be written as a linear combination of training samples. From w_0 we can find that only training samples x_i with Lagrangian multipliers $\alpha_i > 0$ take effect in the sum Eq. (2.34), all training samples with $\alpha_i = 0$ play no role since the combination of the training sample and Lagrangian multiplier is zero.

Substituting Eq. (2.34) and Eq. (2.33) into Eq. (2.31) we can get

$$\begin{aligned} W(\Lambda) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} w_0 \cdot w_0 \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j \end{aligned} \tag{2.35}$$

We can also write this equation in vector notation as

$$W(\Lambda) = \Lambda^T \cdot \mathbf{n} - \frac{1}{2}\Lambda^T D \Lambda \tag{2.36}$$

where **n** is an n dimensional unit vector, and D is a symmetric $n \times n$ matrix with elements

$$D_{ij} = y_i y_j x_i x_j \tag{2.37}$$

To find the desired saddle point we need to locate the maximum of Eq. (2.36) under the constraints Eq. (2.32) and Eq. (2.33) with $\Lambda^T \cdot Y = 0$, where $\Lambda \geq 0$ and $Y^T = (y_1, y_2, \ldots, y_n)$.

The Kuhn-Tucker theorem plays an important role in the theory of optimisation. According to this theorem, at the saddle point in w_0, b_0, Λ_0, any Lagrangian multiplier α_i^0 and its corresponding constraint are connected by an equality

$$\alpha_i[y_i(w_0 \cdot x_i + b_0) - 1] = 0,\ i = 1, 2, \ldots, n \tag{2.38}$$

From this equality we find that non-zero value α_i are only achieved in cases where

$$y_i(w_0 \cdot x_i + b_0) - 1 = 0 \tag{2.39}$$

In other words, $\alpha_i \neq 0$ only stands for cases where the inequality is met as an equality. We call vectors x_i for which $y_i(w_0 \cdot x_i + b_0) = 1$ for *support vectors*. Note that in this terminology the Eq. (2.34) states that the solution vector w_0 can be expanded on support vectors.

Another observation based on the Kuhn-Tucker Eq. (2.32) and Eq. (2.33) for the optimal solution, is the relationship between the maximal value $W(\Lambda_0)$ and the separation distance ρ_0:

$$w_0 \cdot w_0 = \sum_{i=1}^{n} \alpha_i^0 y_i w_0 \cdot x_i = \sum_{i=1}^{n} \alpha_i^0 (1 - y_i b_0) = \sum_{i=1}^{n} \alpha_i^0 \tag{2.40}$$

Substituting this equality into the expression for $W(\Lambda_0)$ we obtain

$$W(\Lambda_0) = \sum_{i=1}^{n} \alpha_i^0 - \frac{1}{2} w_0 \cdot w_0 = \frac{1}{2} \|w_0\|^2 \tag{2.41}$$

The algorithm described previously constructs hyperplane in the input space. To construct a hyperplane in a feature space, one must first transform the n dimensional input vector x into an N dimensional feature vector through a choice of an N dimensional vector function $\phi : \Re^n \to \Re^N$.

An N dimensional linear separator w and a bias b is then constructed for the set of transformed vectors

$$\phi(x_i) = \phi_1(x_i), \phi_2(x_i), \ldots, \phi_N(x_i),\ i = 1, 2, \ldots, n \tag{2.42}$$

Classification of an unknown vector x is done by first transforming the vector to the separating space $x \to \phi(x)$ and then taking the sign of the function

$$g(x) = w \cdot \phi(x) + b \tag{2.43}$$

According to the properties of the soft margin classifier method, the vector w can be written as a linear combination of support vectors in the feature space. That means

$$w = \sum_{i=1}^{n} y_i \alpha_i \phi(x_i) \tag{2.44}$$

The linearity of the dot product implies that the classification function g in Eq. (2.43) for an unknown vector x only depends on the dot product

$$g(x) = w \cdot \phi(x) + b = \sum_{i=1}^{n} y_i \alpha_i \phi(x) \phi(x_i) + b \tag{2.45}$$

The idea of constructing support vector networks comes from considering general forms of the dot product in a Hilbert space

$$\phi(u) \cdot \phi(v) \equiv K(u, v) \tag{2.46}$$

According to the Hilbert-Schmidt Theory, any symmetric function $K(u, v) \in L_2$, can be expanded in the form

$$K(u, v) = \sum_{i=1}^{\infty} \lambda_i \phi_i(u) \cdot \phi_i(v) \tag{2.47}$$

where $\lambda_i \in \Re$ and ϕ_i are eigenvalues and eigenfunctions

$$\int K(u, v) \phi_i(u) du = \lambda_i \phi_i(v) \tag{2.48}$$

of the integral operator defined by the kernel $K(u, v)$. A sufficient condition to ensure that Eq. (2.46) defines a dot product in a feature space is that all the eigenvalues in the expansion Eq. (2.47) are positive. To guarantee that these coefficients are positive, it is necessary and sufficient that the condition

$$\int \int K(u, v) g(u) g(v) du dv > 0 \tag{2.49}$$

is satisfied for all g so that

$$\int g^2(u) du < \infty \tag{2.50}$$

Therefore, functions that satisfy Mercer's theorem can be used as dot product. Aizerman, Braverman and Rozonoer considered a convolution of the dot product in the feature space given by function of the form

$$K(u,v) = exp(-\frac{|u-v|}{\sigma}) \tag{2.51}$$

which they call *Potential Functions* [13].

However, the convolution of the dot product in feature space can be given by any function satisfying Mercer's condition, in particular, to construct a polynomial classifier of degree d in n dimensional input space one can use the following function

$$K(u,v) = (u \cdot v + 1)^d \tag{2.52}$$

Using different dot products $K(u,v)$, one can construct different learning machines with arbitrary types of decision surfaces. The decision surface of these machines has a form

$$g(x) = \sum_{i=1}^{n} y_i \alpha_i K(x,x_i) \tag{2.53}$$

where x_i is the image of a support vector in input space and α_i is the weight of a support vector in the feature space.

To find the vectors x_i and weights α_i, one follows the same solution scheme as for the original optimal margin classifier or soft margin classifier. The only difference is that instead of matrix D, one uses the matrix

$$D_{ij} = y_i y_j K(x_i,x_j),\ i,j = 1,2,\ldots,n \tag{2.54}$$

Thereafter, we give the definition of geometric margin as $\delta_i = \frac{1}{\|w\|}|g(x)|$, that we can see clearly in Fig. 2.20. H is the hyperplane we want to find, H_1 and H_2 are two parallel lines going through samples that are the nearest to H, and the distance between H and H_1 or H_2 is the geometric margin.

The reason why we should choose the maximum margin is that in statistical learning, there is a relationship between geometric margin and misclassification: $Misclassification\ Rate \leq (\frac{2R}{\delta})$, where δ means distance between samples and hyperplane. $R = \max \|x_i\|$ means max length of samples x_i, in other words, how wide the samples spread.

From the formula we can see that the upper bound is determined by geometric margin. The larger the geometric margin is, the smaller the upper bound is.

If we want to find the maximum of the geometric margin, it is equal to finding the minimum of $\|w\|$. So now we have our objective function as $\min \frac{1}{2}\|w\|^2$ (the $\frac{1}{2}$ and square

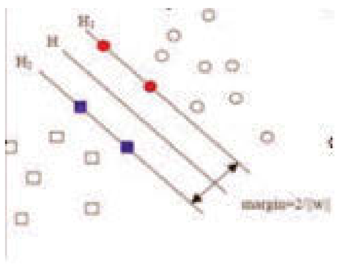


Fig. 2.20 Geometric margin and hyperplane

are both for later convenience). We can see the direct solution is that $\|w\| = 0$ which makes no sense. We get this result only because we haven't thought about the constraints that the data must not lie between H_1 and H_2. So the real expression of binary SVM will be $\min \frac{1}{2}\|w\|^2, s.t.\ y_i \cdot (w^T \phi(x_i) + b) \geq 1$.

We usually see the kernel function defined as

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{2.55}$$

which takes the form of a quadratic programming problem in which we optimise a quadratic function subject to a set of inequality constraints.

Some commonly used kernel functions are: Linear kernel $K(x_i, x_j) = x_i^T x_j$; Polynomial kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d,\ \gamma > 0$; RBF kernel $K(x_i, x_j) = exp(-\gamma\|x_i - x_j\|^2),\ \gamma > 0$; Sigmoid tanh kernel $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r),\ \gamma > 0$.

There are two problems now: the first is which kernel function would we choose for specific problems. The second is what if the problem is still non-linearly separated after using the kernel trick? There is no criterion for choosing the kernel function. Basically, the RBF kernel function usually gives good performance. For the second problem, we will use *slack variables* to modify this approach so that data points are allowed to be on the wrong side of the margin boundary, but with a penalty that increases with the distance from the boundary.

Assume we have another sample on one side of the boundary stated in Fig. 2.21, and the yellow square is on the wrong side. It means the framework is sensitive to outliers. To allow some samples not to satisfy the "hard margin" in the training set, we introduce *slack variable* and propose to optimize a soft-margin criterion. This version of SVM is called "soft-margin" SVM.

Consider the case where the training data cannot be separated without error. In this case one may want to separate the training data with a minimal number of errors. Let us introduce some non-negative variables $\xi_i \geq 0, i = 1, 2, \ldots, n$ to express this formally.

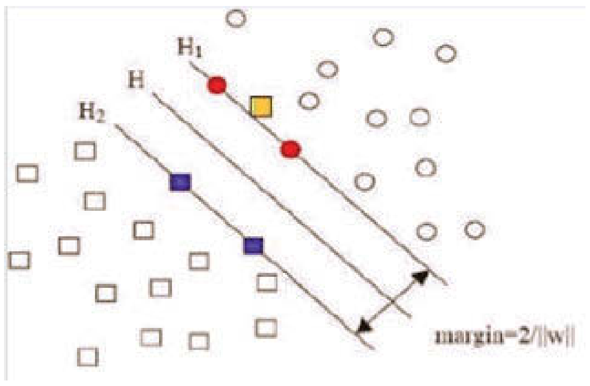


Fig. 2.21 Margin with outlier

We can now minimise the functional

$$\Phi(\xi) = \sum_{i=1}^{n} \xi_i^{\sigma} \tag{2.56}$$

for small $\sigma > 0$, subject to the constraints

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i,\ i = 1, 2, \ldots, n,\ \xi_i \geq 0 \tag{2.57}$$

For sufficiently small σ, Eq. (2.56) describes the amount of training error.

There are at least two ways of introducing slack variables. One may introduce a single slack variable ξ_i for violations of the non-linear constraints (i.e. every instance x_i) [22] or one may penalize margin violations for every linear constraint (i.e. every instance x_i and output $y \neq y_i$) [44]. Since the former will result in a tighter upper bound on the empirical risk and offers some advantages in the proposed optimization scheme, we have focused on this formulation. Adding a penalty term that is linear in the slack variables to the objective, results in the quadratic program:

$$\begin{gathered} \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ s.t. \forall i, \forall y \in Y \backslash y_i : \langle w, \delta\phi_i(y) \rangle \geq 1 - \xi_i, \xi_i \geq 0 \end{gathered} \tag{2.58}$$

Alternatively, we can also penalize margin violations by a quadratic term, leading to the following optimization problem:

$$\begin{aligned} &\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{2n}\sum_{i=1}^{n}\xi_i^2 \\ &s.t. \forall i, \forall y \in Y \backslash y_i : \langle w, \delta\phi_i(y)\rangle \geq 1 - \xi_i \end{aligned} \tag{2.59}$$

In both cases, $C > 0$ is a constant that controls the trade-off between training error minimization and margin maximization.

2.6.2 Multi-class SVM

SVM is fundamentally a binary classifier. In practice, however, we often have to tackle problems involving $K > 2$ classes, which are also called *multi-class SVM*. How to effectively extend the binary SVM for multi-class classification is still an on-going research problem. While some authors have proposed methods that consider all classes at once, the dominant approach for solving multi-class SVM is to reduce a single multi-class SVM problem into multiple binary SVM problems.

The formulation to solve the multi-class SVM problem in one step has variables proportional to the number of classes. Therefore, for multi-class SVM methods, either several binary classifiers have to be constructed or a larger optimisation problem is necessary. In general it is computationally more expensive to solve a multi-class problem than a binary problem with the same number of data.

One-versus-the-rest Approach

One commonly used approach, probably the earliest one for SVM multi-class classification, is to construct k separate SVM models, where k is the number of classes, in which the ith model $y_i(x)$ is trained using the data from class C_i as the positive examples and the data from the remaining $k-1$ classes as the negative examples. This is known as the *one-versus-the-rest* approach [13].

Therefore, given n training data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i \in \Re^n, i = 1, 2, \ldots, n$ and $y_i \in 1, 2, \ldots, k$ is the class of x_i. The ith SVM solves the following problem:

$$\begin{aligned}
&\min_{w^i,b^i,\xi^i} \frac{1}{2}\|w^i\|^2 + \frac{C}{n}\sum_{j=1}^{n}\xi_j^i s.t. \\
&(w^i)^T\phi(x_j) + b^i \geq 1 - \xi_j^i,\ if y_j = i \\
&(w^i)^T\phi(x_j) + b^i \leq -1 + \xi_j^i,\ if y_j \neq i \\
&\xi_j^i \geq 0,\ j = 1,2,\ldots,n
\end{aligned} \tag{2.60}$$

where the training data x_i are mapped to a higher dimensional space by the function ϕ.

Minimising $\frac{1}{2}\|w^i\|^2$ means that we would like to maximise $\frac{2}{\|w^i\|}$, the margin between two groups of data. When data are not linearly separable, there is a penalty term $\frac{C}{n}\sum_{j=1}^{n}\xi_j^i$ which can reduce the number of training errors. The basic concept behind SVM is to search for a balance between the regularisation term $\frac{1}{2}\|w^i\|^2$ and the training errors.

After solving Eq. (2.60), there are k decision functions:

$$\begin{gathered}
(w^1)^T\phi(x) + b^1 \\
\vdots \\
(w^k)^T\phi(x) + b^k
\end{gathered} \tag{2.61}$$

We say x is in the class which has the highest value of the decision function:

$$class\ of\ x \equiv \operatorname*{argmax}_{i=1,2,\ldots,k}((w^i)^T\phi(x) + b^i) \tag{2.62}$$

In practice, we solve the dual problem of Eq. (2.60) whose number of variables is the same as the number of data in Eq. (2.60). Hence k of n-variable quadratic programming problems are solved.

The disadvantage of *one-versus-the-rest* SVM is that the individual classifiers can lead to inconsistent results in which an input is assigned to multiple classes simultaneously, or none. This problem is sometimes addressed by making predictions for new inputs x using $y(x) = \max_k y_k(x)$. Unfortunately, this heuristic approach suffers the problem that the different classifiers were trained on different tasks, and there is no guarantee that the real-valued quantities $y_k(x)$ for different classifiers will have appropriate scales.

The other problem with the *one-versus-the-rest* approach is that the training sets are not balanced. For instance, if we have ten classes each with equal numbers of training data points, then the individual classifiers are trained on data sets comprised of 90% negative examples and only 10% positive examples, and the symmetry of the original problem is lost.

One-versus-one Approach

Another major approach is to train $k(k-1)/2$ different binary class SVMs on all possible pairs of classes, and then to classify test points according to which class has the highest number of "votes", an approach that is sometimes called *one-versus-one* [37]. For training data from the ith and the jth classes, we solve the following binary classification problem:

$$\begin{aligned}
&\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}\|w^{ij}\|^2 + \frac{C}{n}\sum_t \xi_t^{ij} s.t. \\
&(w^{ij})^T\phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij},\ if y_t = i \\
&(w^{ij})^T\phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij},\ if y_t = j \\
&\xi_t^{ij} \geq 0
\end{aligned} \tag{2.63}$$

There are different methods for doing the future testing after all $k(k-1)/2$ classifiers are constructed. The "voting" strategy is the most commonly used method whereby if there is a $\text{sign}((w^{ij})^T\phi(x) + b^{ij})$ say x is in the ith class, then the vote for the ith class is added by one. Otherwise, the jth class is increased by one. Then we predict that x is in the class with the largest vote. The voting strategy described above is also called the "Max Wins" approach. In case that two classes have the same votes, though it may not be a good strategy, we simply select the one with the smaller index.

In practice, we solve the dual of Eq. (2.63) whose number of variables is the same as the number of data in two classes. Hence if on average each class has n/k data points, we have to solve $k(k-1)/2$ quadratic programming problems where each of them has about $2n/k$ variables.

This can lead to ambiguities in the resulting classification. Also, for large k this approach requires significantly more training time than the *one-versus-the-rest* approach. Similarly, to evaluate test points, much more computation is required.

DAGSVM Approach

The other commonly used method for multi-class SVM is called the Directed Acyclic Graph SVM (DAGSVM) [101]. A DAG is a graph whose edges have an orientation and no cycles. A rooted DAG has a unique node that is the only node which has no arcs pointing to it.

To evaluate a particular DAG on input x, starting at the root node, the binary function at a node is evaluated. The node is then exited via the left edge or right depending on the output value. The next node's binary function is then evaluated. The value of the decision function is the value associated with the final leaf node (see Fig. 2.22 for an example of 4 classes DAG). The path taken through the DAG is known as the evaluation path. The input x reaches a node of the graph, if that node is on the evaluation path for x. We refer to the decision node distinguishing classes i and j as ij-node. Assuming that the number of a leaf is its class, this node is the ith node in the $(n-j+i)$th layer provided $i<j$. Similarly the j-nodes are those nodes involving class j, that is, the internal nodes on the two diagonals containing the leaf labeled by j.

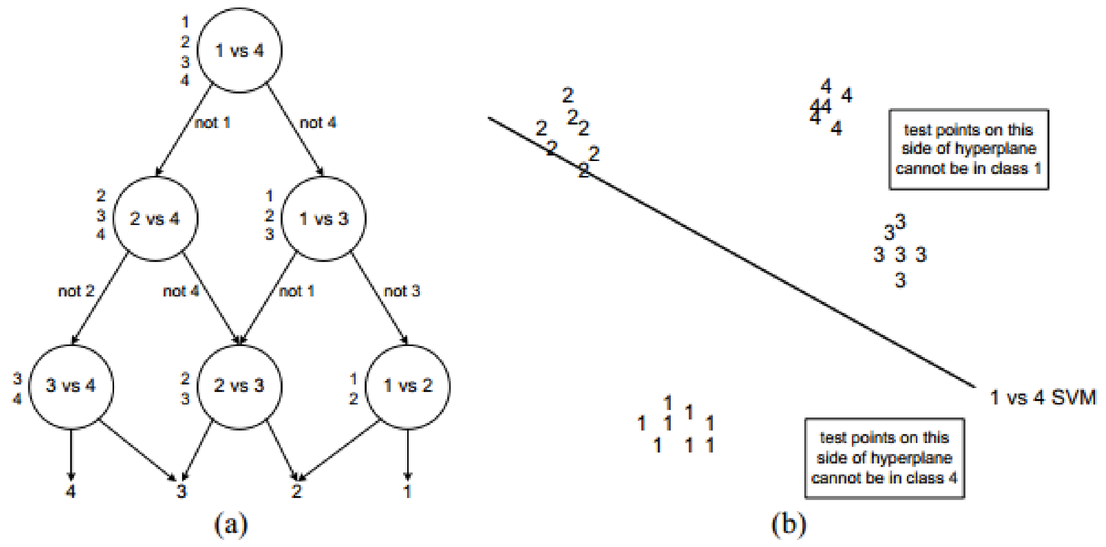

1 vs 4
not 1
not 4
2 vs 4
1 vs 3
not 2
not 4
not 1
not 3
3 vs 4
2 vs 3
1 vs 2
(a)
test points on this side of hyperplane cannot be in class 1
1 vs 4 SVM
test points on this side of hyperplane cannot be in class 4
(b)

Fig. 2.22 (a) The decision DAG for finding the best class out of 4 classes. The equivalent list state for each node is shown next to that node. (b) A diagram of the input space of a 4-class problem. A one-versus-one SVM can only exclude one class from consideration.

The DAG is equivalent to operating on a list, where each node eliminates one class form the list. The list is initialised with a list of all classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the other class is eliminated from the list, and DAG proceeds to test the first and last elements for the new list. The DAG terminates when only one class remains in the list. For a problem with k classes, $k-1$ decision nodes will be evaluated in order to derive an answer. In total, the DAGSVM will contain $k(k-1)/2$ nodes, each with an associated one-versus-one classifier.

An advantage of using a DAG is that [101] some analysis of generalisation can be established. There are still no similar theoretical results for one-versus-the-rest and one-versus-one methods yet. In addition, its testing time is less than the one-versus-one method. Since the number of classifiers of DAGSVM is the same as the it of one-versus-one approach, they share the same disadvantages as well.

One Single Optimisation SVM

Weston proposed an approach for multi-class problems by solving one single optimization problem [157]. The idea is similar to the *one-versus-the-rest* approach. k models where the mth function $w_m^T\phi(x)+b$ which assigns a score to each class are constructed. Given that the models are jointly built, their scales are consistent. This approach solves an optimization problem formulated as:

$$\begin{aligned}
&\min_{w,b,\xi} \frac{1}{2}\sum_{m=1}^{k} \|w_m\|^2 + \frac{C}{n}\sum_{i=1}^{n}\sum_{m\neq y_i} \xi_i^m \quad s.t. \\
&w_{y_i}^T\phi(x_i)+b_{y_i} \geq w_m^T\phi(x_i)+b_m+2-\xi_i^m, \\
&\xi_i^m \geq 0,\ \forall i=1,\ldots,n,\ m\in\{1,\ldots,k\}\backslash y_i
\end{aligned} \tag{2.64}$$

The decision function is

$$\operatorname*{argmax}_{m=1,\ldots,k}(w_m^T\phi(x)+b_m) \tag{2.65}$$

which is the *one-versus-the-rest* method. Yet, this time the scale of the models is consistent.

Also, Crammer and Singer proposed an approach for multi-class problems by solving one single optimization problem [22]. The Crammer and Singer's multi-class SVM is the one sometimes denoted as simply M-SVM. This approach solves an optimization problem formulated as:

$$\begin{aligned}
&\min_{w,\xi} \frac{1}{2}\sum_{m=1}^{k} \|w_m\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \quad s.t. \\
&w_{y_i}^T\phi(x_i)+b_{y_i} - (w_m^T\phi(x_i)+b_m) \geq e_i^m - \xi_i, \\
&\forall i=1,\ldots,n
\end{aligned} \tag{2.66}$$

where $e_i^m = 1-\delta_{y_i,m}$ and δ is the Kronecker delta function. The decision function is again

$$\operatorname*{argmax}_{m=1,\ldots,k}(w_m^T\phi(x)+b_m) \tag{2.67}$$

2.6.3 Structural SVM

We only consider the input and output data as i.i.d. in the previous section, however, in reality, we often encounter problems that the output is structured models such as sequences, strings, trees, lattices or graphs. Such problems arise in a variety of applications, ranging from multi-label classification and classification with class taxonomies, to label sequence learning, sequence alignment learning, and supervised grammar learning, natural language parsing, to name just a few. The general problem is to learn a mapping from input vector or patterns $x \in X$ to discrete response variables $y \in Y$, based on a training sample of input-output pairs $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ drawn from some fixed but unknown probability distribution. Unlike multi-class classification, where the output space consists of an arbitrary finite set of labels or class identifiers $Y = 1, 2, \dots, k$, we now consider the case where elements of Y are structured objects.

Tsochantaridis et al. proposed an approach designing a classification algorithm that can deal with more complex outputs [134]. More generally, they consider problems involving multiple dependent output variables, structured output spaces by generalising large margin methods, more specifically multi-class SVM [22, 157], to the broader problem of learning structured response. The naive approach of treating each structure as a separate class is often intractable, since it leads to a multi-class problem with a very large number of classes. They overcome this problem by specifying discriminant functions that exploit the structure and dependencies within Y.

The general problem is to learn functions $f: X \to Y$ between input spaces X and arbitrary discrete output spaces Y based on a training sample of input-output pairs. Consider the case of natural language parsing, where the function f maps a given sentence x to a parse tree y. This is depicted graphically in Fig. 2.23.

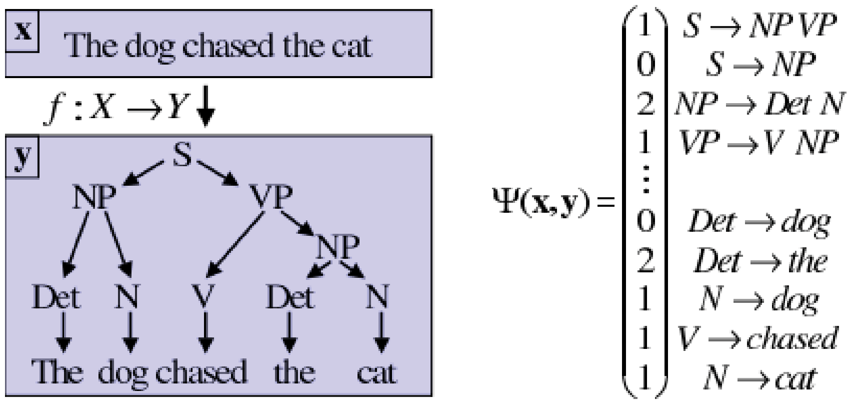


Fig. 2.23 Illustration of natural language parsing model

The approach is to learn a discriminant function $F: X \times Y \to \Re$ over input-out pairs from which a prediction by maximizing F over the response variable for a specific given input x. Hence, the general form of hypotheses f is

$$f(x;w) = \underset{y \in Y}{\operatorname{argmax}} F(x,y;w) \tag{2.68}$$

where w denotes a parameter vector.

Assume F to be linear in some combined feature representation of inputs and outputs $\phi(x,y)$, i.e.

$$F(x,y;w) = \langle w, \phi(x,y) \rangle \tag{2.69}$$

The specific form of ψ depends on the nature of the problem and special cases will be discussed subsequently. For instance, in the case of multi-class SVM, $\langle w, \phi(x,y) \rangle = w_y^T \phi(x) + b_y$.

Using the natural language parsing as an illustrative example, we can choose F such that we get a model that is isomorphic to a probabilistic context free grammar. Each node in a parse tree y for a sentence x corresponds to grammar rule g_j, which in turn has a score w_j. All valid parse trees y (i.e. trees with a designated start symbol S as the root and the words in the sentence x as the leaves) for a sentence x are scored by the sum of the w_j of their nodes. This score can thus be written in the form of Eq. (2.69), with $\phi(x,y)$ denoting a histogram vector of counts (how often each grammar rule g_j occurs in the tree y). $f(x;w)$ can be efficiently computed by finding the structure $y \in Y$ that maximizes $F(x,y;w)$ via the CKY algorithm [81].

Using the natural language parsing as an illustrative example, we can choose F such that we get a model that is isomorphic context free grammar. Each node in a parse tree y for a sentence x corresponds to grammar rule g_j, which in turn has a score w_j. All valid parse tree y (i.e. trees with a designated start symbol S as the root and the words in the sentence x as the leaves) for a sentence x are scores by the sum of the w_j of their nodes. This score can thus be written in the form of $F(x,y;w) = \langle w, \psi(x,y) \rangle$ with $\psi(x,y)$ denoting a histogram vector of counts (how often each grammar rule g_j occurs in the tree y). $f(x;w)$ can be efficiently computed by finding the structure $y \in Y$ that maximizes $F(x,y;w)$ via the CKY algorithm [20].

The standard 0-1 loss function typically used in classification is not proper for most structured responses. Take natural language parsing as an example, a parse tree that is almost correct and has only one or a few different nodes from the correct parse should be given a much less loss from a parse tree that is completely different.

In order to quantify the accuracy of a prediction, consider learning with arbitrary loss function $\Delta(y,\bar{y})$ quantifying the loss associated with a prediction $\bar{y}$, if the ground truth output value is y. Here $\Delta(y,y) = 0$ and $\Delta(y,\bar{y} > 0)$, $\forall y \neq \bar{y}$.

The goal is to find a function f in a given hypothesis class such that the risk $R_P^{\triangle}(f) = \int_{X\times Y}\Delta(y, f(x))dP(x,y)$ is minimized. P is unknown and following the supervised learning paradigm.

Margin Maximisation

First, consider the case where there exists a function f parameterized by w such that the empirical risk is zero. The condition of zero training error can then be compactly written as a set of non-linear constraints.

$$\forall i \in \{1,\dots,n\} : \; max_{y\in Y\backslash y_i}\{\langle w, \phi(x_i,y)\rangle\} < \langle w, \phi(x_i,y_i)\rangle \tag{2.70}$$

Every one of the non-linear inequalities can be equivalently replaced by $|Y|-1$ linear inequalities, resulting in a total of $n\cdot|Y|-n$ linear constraints,

$$\forall i \in \{1,\dots,n\}, \; \forall \in Y\backslash y_i, \; \langle w, \phi(x_i,y_i)\rangle - \langle w, \phi(x_i,y)\rangle \geq 0 \tag{2.71}$$

Here we also define $\delta\phi_i(y) \equiv \phi(x_i,y_i) - \phi(x_i,y)$, so that the constraints can be more compactly written as $\langle w, \delta\phi_i(y)\rangle \geq 0$.

If the set of inequalities of the former is feasible, there will typically be more than one solution w^*. To specify a unique solution, w is selected with $\|w\| \leq 1$ for which the score of the correct label y_i is uniformly most different from the closest runner-up $\bar{y}_i(w) = \text{argmax}_{y\neq y_i}\langle w, \phi(x_i,y)\rangle$. This generalizes the maximum margin principle employed in SVM to the more general case considered in our research. The resulting hard margin optimization problem can be expressed as a convex quadratic program in the form:

$$\begin{aligned} &SVM_0 : \; \min_w \frac{1}{2}\|w\|^2 \\ &\forall i, \forall y \in Y\backslash y_i : \; \langle w, \delta\phi_i(y)\rangle \geq 1 \end{aligned} \tag{2.72}$$

To allow errors in the training set, introduce slack variables and propose to optimize a soft margin criterion. Adding a penalty term that is linear in the slack variables to the objective results in the quadratic program:

$$\begin{aligned} &SVM_1 : \; \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &s.t. \; \forall i, \forall y \in Y\backslash y_i : \; \langle w, \delta\phi_i(y)\rangle \geq 1-\xi_i, \xi_i \geq 0 \end{aligned} \tag{2.73}$$

Alternatively, penalize margin violations by a quadratic term $\frac{C}{2n}\sum_i \xi_i^2$, giving another soft-margin solution:

$$\begin{aligned} &SVM_2: \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{2n}\sum_{i=1}^{n}\xi_i^2 \\ &s.t.\ \forall i, \forall y \in Y\backslash y_i: \ \langle w, \delta\phi_i(y)\rangle \geq 1-\xi_i, \xi_i \geq 0 \end{aligned} \tag{2.74}$$

For the case of arbitrary loss functions, re-scale the slack variables according to the loss incurred in each of the linear constraints. Intuitively, violating a margin constraint involving a $y \neq y_i$ with hinge loss $\Delta(y, y_i)$ should be penalized more severely than a violation involving an output value with smaller loss. This can be accomplished by multiplying the margin violation by the loss, or equivalently by scaling the slack variable with the inverse loss, which gives a justification of soft margin as:

$$\begin{aligned} &SVM_1^{\Delta s}: \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &s.t.\ \forall i, \forall y \in Y\backslash y_i: \ \langle w, \delta\phi_i(y)\rangle \geq 1-\frac{\xi_i}{\Delta(y_i,y)} \end{aligned} \tag{2.75}$$

where we have given that $\Delta(y_i, y) > 0$ for $y_i \neq y$ and $\Delta(y, y) = 0$.

The optimization problem $SVM_2^{\Delta s}$ can be derived where $\Delta(y_i, y)$ is replaced by $\sqrt{\Delta(y_i, y)}$.

In addition to the *slack re-scaling* approach, a second way to include loss functions is to re-scale the margin. This method was first proposed by Taskar et al. [116] for the special case of the Hamming loss, but can be straightforward to general loss functions. The quadratic problem is expressed as:

$$\begin{aligned} &SVM_1^{\Delta m}: \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &s.t.\ \forall i, \forall y \in Y: \ \langle w, \delta\phi_i(y)\rangle \geq \Delta(y_i,y)-\xi_i \end{aligned} \tag{2.76}$$

Similarly, replace $\Delta(y_i, y)$ by $\sqrt{\Delta(y_i, y)}$ then we can get $SVM_2^{\Delta m}$.

Tsochantaridis et al. have also shown the advantages and disadvantages of the two formulations. An appealing property of the slack re-scaling approach is its scaling invariance. On the other hand, the margin re-scaling formulation is not invariant under scaling of the loss function. For example, one needs to re-scale the feature map ϕ by a corresponding scale factor as well. This seems to indicate that one has to calibrate the scaling of the loss and the scaling of the feature map more carefully in the $SVM_1^{\Delta m}$ formulation. The $SVM_1^{\Delta s}$ formulation however, represents the loss scale exactly in terms of the constant C.

In practice, in the literature the margin-rescaling formulation has proved the most popular, and we will follow it in our experiments.

Given a set of N training instances $\{x^i, y^i\}$, $i = 1, \ldots, N$, structural SVM finds the optimal model's parameter vector w by solving the following convex optimisation problem:

$$\begin{aligned} &\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi^i \quad s.t. \\ &w^\top \phi(x^i, y^i) - w^\top \phi(x^i, y) \geq \Delta(y^i, y) - \xi^i, \\ &\qquad i = 1 \ldots N, \ \forall y \in \mathscr{Y} \end{aligned} \tag{2.77}$$

As usual, term $\sum_{i=1}^{N} \xi^i$ places an upper bound over the total training error, while term $\|w\|^2$ regularises the solution to encourage generalisation. Parameter C is an arbitrary, positive coefficient that balances these two terms. In the constraints, function $\phi(x, y)$ is a feature function that computes structured features from the pair $\{x, y\}$ such that $w^\top \phi(x, y)$ can assign a score to the pair. The constraint for labeling $y = y^i$ guarantees that $\xi^i \geq 0$. Eventually, $\Delta(y^i, y)$ is the chosen, arbitrary loss function.

The problem with Eq. (2.77) is that the size of the constraint set, $\mathscr{Y}$, is exponential in the number of of the output variables and it is therefore impossible to satisfy the full constraint set. However, [134] has shown that it is possible to find ε-correct solutions with a constraint subset of polynomial size, consisting of only the "most violated" constraint for each sample, i.e. the labeling with the highest sum of score and loss:

$$\begin{aligned} &\xi^i = \max_y(-w^\top \phi(x^i, y^i) + w^\top \phi(x^i, y) + \Delta(y^i, y)) \\ &\rightarrow \bar{y}^i = \operatorname*{argmax}_y(w^\top \phi(x^i, y) + \Delta(y^i, y)) \end{aligned} \tag{2.78}$$

This problem is commonly referred to as "loss-augmented inference" due to its resemblance to the usual inference of Eq. (3.5) and is the main step of structural SVM.

Algorithm for Structural SVM

Now the key challenge is that the size of each quadratic problem is intractable, since there will be $n|Y| - n$ margin inequalities. In many cases, $|Y|$ may be extremely large. Suppose a simple case in sequential labeling, each state can be chosen from 5 classes, and the length of the sequence is 100. The value of $|Y|$ is $5^{100} \approx 10^{69}$. Many other cases have shown that the cardinality may grow exponentially in the description length of y, which makes standard quadratic programming solvers unsuitable for this type of problem.

In [134], the authors have proposed an algorithm that exploits the special structure of the maximum margin problem, so that only a much smaller subset of constraints needs to be explicitly examined. The algorithm can compute arbitrary close approximation in polynomial time for a large range of structures and loss functions. This is a valid strategy, since there always exists a polynomial sized subset of constraints so that the solution of the relaxed problem from the original optimization problem is up to a precision of ε. This means that most constraints are guaranteed to be violated by not more than ε, without the need to add all constraints to the optimization problem. Since we will also use this algorithm later in the research, we hereby give a detailed description.

The algorithm maintains working set S_i for each training example and proceeds by finding the "most violated" constraints for x_i, involving some output value $\bar{y}$ through iterating the training examples (x_i, y_i). If the margin violation of this constraint exceeds the current value of ξ_i by more than ε, the dual variable corresponding to $\bar{y}$ is added to the working set. Once a constraint has been added, the solution is re-computed with respect to S. Notice that all variables that are not included in the respective working set are treated as 0. The algorithm stops if no constraint is violated by more than ε. Pseudo-code of the algorithm can be found in Algorithm 3.

A convenient property of the cutting plane algorithm is that it has a general and well-defined interface independent of the choice of ϕ and Δ. To apply the algorithm, it is sufficient to implement the feature mapping $\phi(x,y)$, the loss function $\Delta(y_i,y)$ and the maximisation step in the algorithm. While the modeling of $\phi(x,y)$ and $\Delta(y_i,y)$ is typically straightforward, solving the maximisation problem for the "most violated" constraint only requires exploiting the structure of ϕ for output spaces which cannot be dealt with by exhaustive search.

Replace the loss function with a piecewise linear convex upper bound, $\Delta(y_i, y'(w)) \leq \max_{y' \in Y}[\Delta(y_i,y') + w^T\phi(x_i,y')] - w^T\phi(x_i,y_i)$, where $y'(w) = \operatorname{argmax}_{y \in Y} w^T\phi(x_i,y)$.

Instead of minimizing the true empirical risk with the margin maximization, structural SVM solves a convex optimization problem. This problem can be formulated as

$$\min_w \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\max_{y' \in Y}[\Delta(y_i,y') + w^T\psi(x_i,y') - w^T\psi(x_i,y_i)] \tag{2.80}$$

Algorithm 1 Algorithm for computing SVM_0 and the loss re-scaling formulations SVM_1^* and SVM_2^*.

```
**Input**: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n), C, \varepsilon$
$S_i \leftarrow \emptyset$ for all $i = 1, 2, \dots, n$
**repeat**
  **for** $i = 1, 2, \dots, n$ **do**
    /*prepare loss function for optimisation*/
    set up loss function
      $$H(y) \equiv \begin{cases} 1 - \langle w, \delta\phi_i(y) \rangle & (SVM_0) \\ (1 - \langle w, \delta\phi_i(y) \rangle)\Delta(y_i, y) & (SVM_1^{\Delta s}) \\ \Delta(y_i, y) - \langle w, \delta\phi_i(y) \rangle & (SVM_1^{\Delta m}) \end{cases} \quad (2.79)$$
     where $w \equiv \sum_j \sum_{y' \in S_j} \alpha_{jy'} \delta\phi_j(y')$
    /*find cutting plane*/
    compute $\bar{y} = \mathrm{argmax}_{y \in Y} H(y)$
    /*determine value of current slack variable*/
    compute $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
     **if** $H(\bar{y}) > \xi_i + \varepsilon$ **then**
     /*add constraint to the working set*/
     $S_i \leftarrow S_i \cup \{\bar{y}\}$
     **end if**
  **end for**
**until** no $S_i$ has changed during iteration
```

2.6.4 Multi-label Classification with Arbitrary Loss Function

All minimum empirical risk classifiers like SVM aim to minimise a loss function over the training set. For the i.i.d. case, this function is typically the 0-1 loss, but it could be in other cases. Assume we have a known loss function Δ which assigns a non-negative number to every possible pair of outputs. This loss function represents how much we want to penalize a prediction $\bar{y}$ when the ground truth label is y. For example, if we want to use F_1 loss, since F_1 is a score of alignment between y and $\bar{y}$, our choice for the loss is $\Delta(y,\bar{y}) = 1 - F_1(y,\bar{y})$, which is by definition $\Delta(y,\bar{y}) = 1 - \frac{2*y^T*\bar{y}}{Y^T*y+\bar{y}^T*\bar{y}}$.

The next assumption is that the prediction for a given input x returns the maximization of a linear score of the model parameter vector θ, i.e. a prediction is given by $\bar{y}$ such that $\bar{y} = \text{argmax}_{y\in Y}\langle w, \phi(x,y)\rangle$. Here we assume that $\phi(x,y)$ is linearly composed of features of the instances encoded in each v, i.e. $\phi(x,y) = \sum_{v=1}^{V} y_v(\psi_v \otimes x)$. The vector φ_v is the feature representation for the instance v. The map $\phi(x,y)$ will be zero vector whenever $y_v = 0$, i.e. when instance v doesn't have label x.

The next step is to formulate the estimator. An initial, ideal estimator will take the form $w^* = \text{argmin}_w[(\frac{1}{n}\sum_{i=1}^{n}\Delta(\bar{y}^i(x^i;w),y^i)) + \frac{\lambda}{2}\|w\|^2]$. In other words, we want to find a model that minimizes the average prediction loss in the training set *plus* a quadratic regularizer that penalizes complex solutions (the parameter λ determines the trade-off between data fitting and good generalization).

Note the above problem is non-convex. More critically, the loss is a piecewise constant function of w. Here we use an analogous approach, proposed by Tsochantaridis, which optimizes a convex upper bound on the structured loss. The resulting optimization problem is $[w^*,\xi^*] = \text{argmin}_{w,\xi}[\frac{1}{n}\sum_{i=1}^{n}\xi_n + \frac{\lambda}{2}\|w\|^2]$, the constraints would be $\langle w,\phi(x^i,y^i)\rangle - \langle w,\phi(x^i,y)\rangle \geq \Delta(y,y^i) - \xi^i$, $\xi^i \geq 0$. It is easy to see that ξ^* upper bounds $\Delta(\bar{y}^i_*,y^i)$. Here $\bar{y}^i_* \doteq \text{argmax}_y\langle w^*,\phi(x^i,y)\rangle$. First note that since the constraint hold for all y, they also hold for $\bar{y}^i_*$. Second, the left hand side of the inequality for $y = \bar{y}^i$ must be non-positive from the definition of $\bar{y}$. It then follows that $\xi^*_i \geq \Delta(\bar{y}^i_*,y^i)$.

The constraints basically mean a loss sensitive margin. θ is learned so that mispredictions y will end up with a score $\langle w,\phi(x^i,y)\rangle$ that is smaller than the score $\langle w,\phi(x^i,y^i)\rangle$ of the correct prediction y^i by a margin equal to that loss minus slack variables ξ. The formulation is a generalization of support vector machines for the case in which there are an exponential number of classes y.

The optimization problem has $n\cdot|Y| = n\cdot 2^V$ constraints. Naturally, this number is too large to allow for a practical solution of the quadratic program. So we have to construct a constraint generation strategy for selecting a polynomial number of constraints. This method can approximate the original problem particularly well. The constraint generation algorithm consists of starting with no constraints and iteratively adding the most violated constraints for the current solution of the optimization problem. Such an approach is assumed to find an ε-close approximation of the solution after including only $O(\varepsilon^{-2})$ constraints. The key problem that needs to be solved at each iteration is constraint generation which is to find the maximization of the violation margin ξ_i^*, $y_i^* = \text{argmax}_{y \in Y}[\Delta(y, y^i) + \langle w, \phi(x^i, y)\rangle]$. The difficulty in solving the above optimization problem depends on the choice of $\phi(x,y)$ and Δ. We summarise the most commonly used loss functions in the following subsection.

2.6.5 Loss Functions

A loss function, $\Delta(y^i, y)$, quantifies the loss, or cost, carried by a misclassification. In the case of structured labels, a loss function is said to be *decomposable* if $\Delta(y^i, y) = f(\delta(y_t^i, y_t)), t = 1 \dots T$, where $\delta(y_t^i, y_t)$ notes the loss over an individual label. We have mentioned this in Chapter 2 for the 0-1 loss and the Hamming loss, but there are many other loss functions that can be used in different applications. In the following, we review the main loss functions for both the decomposable and non-decomposable cases.

Zero-one loss

The simplest and most commonly used loss function in classification is the *zero-one loss*, assigning a unit loss to every misclassification and a zero loss to every correct classification:

$$\Delta_{0-1}(y^i, y) = \begin{cases} 1 & \text{for } y \neq y^i \\ 0 & \text{for } y = y^i \end{cases} \tag{2.81}$$

In the case of structured labeling, $\Delta_{0-1}(y^i, y) = 1$ if at least one of the individual labels is mispredicted. This shows that the zero-one loss can be evaluated frame by frame and is therefore decomposable.

Hamming loss

Although the zero-one loss is the most common in conventional classification, it is not the most suited for structured prediction since we seek to differentiate the loss for mispredictions with different amounts of wrongly-predicted individual labels. The most common loss function for structured prediction is the *Hamming loss* that accounts for a loss “unit” for every incorrect individual label [100]:

$$\Delta_{Hamming}(y^i, y) = \frac{1}{T}\sum_{t=1}^{T} y_t^i \oplus y_t \tag{2.82}$$

where operator $\oplus$ is the modulo-2 sum or *XOR* boolean operator. Based on this definition, the Hamming loss is also clearly a decomposable case.

Precision at k **and at** r

Although the Hamming loss is suited for structured prediction since it considers graded losses, it does not reflect desirable performance in some important cases. Consider a simple example: the ground truth, y^i, having 3 positive labels and 97 negative labels. In this case, a dummy classifier with constant negative output would carry a loss of only 3%. This asymmetric situation is especially common in detection where positive labels are far less frequent than negative ones. Therefore, performance evaluations must account for positive and negative errors separately. Two useful accuracy measures are the *precision* and *recall*:

$$p = \frac{TP}{TP+FP}; \qquad r = \frac{TP}{TP+FN} \tag{2.83}$$

where TP, FP and FN are the number of true positives, the number of false negatives and the number of false positives, respectively, computed from the classification contingency table of the individual labels of y^i and y. Therefore, the recall assesses the fraction of positive labels correctly predicted by the classifier while the precision measures the amount of true positives amongst all the predicted positives.

In applications where the classifier also produces a ranking of the samples such as in search engines, a user is typically only interested in the top predictions. Therefore, a common performance figure is the precision measured over the top k predictions, known as *precision at k* ($p_{@k}$):

$$p_{@k} = \frac{TP}{TP+FP} \quad s.t. \;\; TP+FP = k \tag{2.84}$$

A similar figure is the precision at a chosen value of recall, $p_{@r}$:

$$p_{@r} = \frac{TP}{TP+FP} \quad s.t. \;\; \frac{TP}{TP+FN} = r \tag{2.85}$$

For figures like the precision at k and the precision at r, the predicted labeling, y, must abide by a constraint. Therefore, the domain is restricted and we note it as $\mathscr{Y}_k$ and $\mathscr{Y}_r$, respectively, in the rest of the thesis.

Since the precision's denominator depends on the entire prediction, the precision is certainly not a decomposable function of the individual labels. Given that it ranges between 0 and 1, the corresponding loss can be conveniently defined as $\Delta_p(y^i, y) = 1 - p$.

Precision/recall break-even point

Assuming that the precision behaves as a non-increasing function of the recall, one can explore the entire recall range between 0 and 1 to search for the value where $p = r$. This value is known as the *precision/recall break-even point* and can be defined as:

$$p_{PRBEP} = \frac{TP}{TP+FP} \quad s.t. \;\; p = \frac{TP}{TP+FN} \;\leftrightarrow\; FP = FN \tag{2.86}$$

Clearly, the higher the value, the better is the combined performance over the two figures. As before, the corresponding loss is defined as $\Delta_{p_{PRBEP}}(y^i, y) = 1 - p_{PRBEP}$.

F_1 loss

Precision and recall effectively describe the labeling performance. However, for performance comparison purposes it would be desirable to have them combined into a single figure. To this end, the F_1 *measure* is introduced as:

$$F_1(y^i, y) = \frac{2pr}{p+r} = \frac{2\,TP}{2\,TP+FN+FP} \tag{2.87}$$

with corresponding loss $\Delta_{F_1}(y^i, y) = 1 - F_1(y^i, y)$. Given that it incorporates the precision, the F_1 loss is also non-decomposable. It is, however, certainly more suited than the Hamming loss to represent performance in asymmetric cases: for instance, in the example above, the F_1 loss of the dummy classifier would be 1 (i.e., complete loss). For this reason, we report and compare performance in this paper using the F_1 measure as the main figure.

Average precision

Another popular accuracy measure is the *average precision* (AP) that is the average of the precision at various levels of recall. This measure is useful since it can assess the labeling performance at different trade offs between false positives and false negatives. Assuming R

represents different levels of recall (typically, all levels between 0 and 1 in 0.1 steps), the average precision is defined as:

$$AP = \frac{1}{R}\sum_{r} p_{@r} \tag{2.88}$$

Everingham *et al.* in [30] have provided an alternative definition for the AP ensuring that the precision behaves as a monotonically non-increasing function of the recall:

$$AP = \frac{1}{R}\sum_{r} \max_{l=0\ldots r} p_{@l} \tag{2.89}$$

This way of computing the average precision has become dominant in computer vision and is therefore adopted in this paper for the experimental evaluation.

For conventional classification, the computation of the average precision can be simplified if the classifier not only provides a prediction, but also ranks all the samples in score order. Unfortunately, in the case of structured prediction, the measurements are scored jointly and cannot be ranked individually. Therefore, the computation of the average precision requires the classifier to explicitly provide the set of predictions at the required recall levels.

As usual, the corresponding loss can be defined as the complement to the measure, $\Delta_{AP} = 1 - AP$. Such a loss is not only non-decomposable, but it also cannot be expressed as a function of the classification contingency table.

Chapter 3

Sequential Labeling With Structural SVM Under the Average Precision Loss

In this chapter, we propose a structural SVM learning algorithm for sequential labeling that maximises an average precision measure. A further contribution is an algorithm that computes the average precision of a sequential classifier at test time, making it possible to assess sequential labeling under this measure. Experimental results over challenging datasets which depict human actions in kitchen scenarios (i.e., TUM Kitchen and CMU Multimodal Activity) show that the proposed approach leads to a significant improvement of average precision.

3.1 Introduction and Related Work

Choosing appropriate performance measures plays an important role in developing effective information retrieval and classification systems. Common figures include the false positive and false negative rates, the precision and recall, and the F-measure which can all assess the accuracy of a prediction by comparing the predicted labels with the given ground-truth labels. However, in applications such as information retrieval, it is often important to assess not only the accuracy of the predicted labels, but also that of a complete ranking of the samples. In classification, too, it is often preferable to evaluate the prediction accuracy at various trade-offs of precision and recall, to ensure coverage of multiple operating points. For both these needs, the average precision (a discretised version of the area under the precision-recall curve) offers a very informative performance measure.

Amongst the various flavours of classification, sequential labeling, or tagging, refers to the classification of each of the measurements in a sequence. It is a very important task in a variety of fields including video analysis, bioinformatics, financial time series and natural language processing [107]. Unlike the classification of independent samples, the typical sequential labeling algorithms such as Viterbi (including their n-best versions [90]) do not provide multiple predictions at varying trade-offs of precision and recall, and therefore the computation of their average precision is not trivial.

In the literature, a number of papers have addressed the average precision as a performance measure in the case of independent samples. For instance, [66] has studied the statistical behaviour of the average precision in the presence of relevance judgements. Yilmaz and Aslam in [165] have proposed an approximation of the average precision in retrieval systems with incomplete and imperfect judgements. Morgan *et al.* in [87] have proposed an algorithm for learning the weights of a search query with maximum average precision. Notably, Joachims *et al.* in [166] have proposed a learning algorithm that can efficiently train a support vector machine (SVM) under an average precision loss. Mani Ranjbar *et al* has proposed a max-margin parameter learning approach for structured prediction problems with non-decomposable performance measures in [111, 110]. However, all this work only considers independent and identically distributed (i.i.d.) samples, while very little work to date has addressed the average precision in sequential labeling and structured prediction. In [118], Rosenfeld *et al.* have proposed an algorithm for training structural SVM under the average precision loss. However, their algorithm assumes that the structured output variables can be ranked in a total order relationship which is generally restrictive.

For the above reasons, we propose a training algorithm that can train structural SVM for sequential labeling under an average precision loss. Our assumptions are very general and do not require ranking of the output space. The core component of our training algorithm is an inference procedure that returns sequential predictions at multiple levels of recall. The same inference procedure can also be used at test time, making it possible to evaluate the average precision of sequential labeling algorithms and to compare it with that of i.i.d. classifiers.

Experiments have been conducted over two challenging sequential datasets: the TUM Kitchen and the CMU-MMAC activity datasets [130, 24]. The results, reported in terms of average precision, show that the proposed method remarkably outperforms other performing classifiers such as standard SVM and structural SVM trained with conventional losses.

3.2 Background

3.2.1 Average Precision

The average precision (AP) is a de-facto standard evaluation in the computer vision community since the popular PASCAL VOC challenges [30]. It is defined as the average of the precision at various levels of recall and is a discretised version of the area under the precision-recall curve (AUC). The AP is a very informative measure since it assesses the classification performance at different trade-offs of precision and recall, reflecting a variety of operating conditions. Its formal definition is:

$$AP = \frac{1}{R}\sum_{r} p_{@r} \tag{3.1}$$

where $p_{@r}$ is the precision at level of recall r, and R is the number of levels. The recall ranges between 0 and 1, typically in 0.1 steps. At its turn, the precision at a chosen value of recall, $p_{@r}$, is defined as:

$$p_{@r} = \frac{TP}{TP+FP} \quad s.t. \;\; \frac{TP}{TP+FN} = r \tag{3.2}$$

where TP, FP and FN are the number of true positives, the number of false negatives and the number of false positives, respectively, computed from the classification contingency table of the predicted and ground-truth labels.

In general, the precision tends to decrease as r grows. However, as defined above, the AP is not a monotonically non-increasing function of r. To ensure monotonicity, Everingham *et al.* in [30] modified its definition as:

$$AP = \frac{1}{R}\sum_{r} \max_{l=0\ldots r} p_{@l} \tag{3.3}$$

This way of computing the average precision has become commonplace in the computer vision and machine learning communities and it is therefore adopted in our experiments. However, the algorithm we describe in Section 3.3 can be used interchangeably for either (3.1) or (3.3). Given that the AP is bounded between 0 and 1, a natural definition for an AP-based loss is $\Delta_{AP} = 1 - AP$.

Sequential labeling predicts a sequence of class labels, $y = (y_1, \ldots, y_t, \ldots, y_T)$, from a given measurement sequence, $x = (x_1, \ldots, x_t, \ldots, x_T)$, where x_t is a feature vector at sequence position t and y_t is a corresponding discrete label, $y_t \in 1 \ldots M$. In many cases, it is not restrictive to assume that y_t is a binary label (1: positive class; 0: negative class), obtaining multi-class classification from a combination of binary classifiers. Therefore, in the following we focus on the binary case. The most widespread model for sequential labeling is the hidden Markov model (HMM) which is a probabilistic graphical model factorising the joint probability of the labels and the measurements. By restricting the model to the exponential family of distributions and expressing the probability in a logarithmic scale, the score of an HMM can be represented as a generalised linear model:

$$\begin{aligned} \ln p(x,y) \propto w^\top \phi(x,y) = w_{init}^\top f(y_1) + \\ + \sum_{t=2}^{T} w_{tran}^\top f(y_t, y_{t-1}) + \sum_{t=1}^{T} w_{em}^\top f(x_t, y_t) \end{aligned} \tag{3.4}$$

where w_{init} are the first-frame parameters, w_{tran} are the transition parameters, and w_{em} are the emission parameters. The inference problem for this model consists of determining the best class sequence for a given measurement sequence:

$$\bar{y} = \underset{y}{\text{argmax}}\ w^\top \phi(x,y)) \tag{3.5}$$

This problem can be efficiently solved in $O(T)$ time by the well-known Viterbi algorithm operating in a logarithmic scale. Viterbi is an instance of dynamic programming where the sequence is scanned in a left-to-right manner and partial solutions are built at every frame. The final solution is guaranteed to be the global optimum [107].

3.3 Training and Testing Sequential Labeling with the AP Loss

The loss functions used for training structural SVM commonly include the 0-1 loss and the Hamming loss. Under these losses, the loss-augmented inference can still be computed by a conventional Viterbi algorithm with adjusted weights. Instead, training with the average precision cannot be approached in the same way since it requires predicting either a ranking or multiple labelings. For this reason, we propose a different formulation of the structural SVM primal problem:

$$
\begin{aligned}
&\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi^i \quad s.t. \\
&w^\top\phi(x^i,y^i) - \frac{1}{R}\sum_r w^\top\phi(x^i,y^{[r]}) \\
&\quad \geq \Delta_{AP}(y^i,y^{[0]},\dots y^{[1]}) - \xi^i, \xi^i \geq 0,\ i = 1\dots N, \\
&\quad r = 0, 0.1, \dots 1,\ \forall y^{[0]}\dots y^{[1]} \in \mathscr{Y}_0 \times \dots \times \mathscr{Y}_1
\end{aligned}
\tag{3.6}
$$

The constraints in Eq. (3.6) state that the score assigned to the ground-truth labeling, y^i, must be greater than or equal to the average score of any set of R labelings at the appropriate levels of recall by at least their average precision loss. In this way, we retain the structural SVM principle of imposing a margin between the ground truth and the prediction that is equal to the chosen loss, while we constrain all the predictions at the prescribed levels of recall.

For Eq. (3.6), the loss-augmented inference becomes:

$$
\begin{aligned}
&\bar{y}^{[0]}\dots\bar{y}^{[1]} = \underset{y^{[0]}\dots y^{[1]}}{\operatorname{argmax}} \left(\frac{1}{R}\sum_r w^\top\phi(x^i,y^{[r]}) + \Delta_{AP}(y^i,y^{[0]},\dots y^{[1]})\right) \\
&= \underset{y^{[0]}\dots y^{[1]}}{\operatorname{argmax}} \left(\frac{1}{R}\sum_r w^\top\phi(x^i,y^{[r]}) + \frac{1}{R}\sum_r \Delta_{p@r}(y^i,y^{[r]})\right) \\
&= \underset{y^{[0]}}{\operatorname{argmax}}(w^\top\phi(x^i,y^{[0]}) + \Delta_{p@0}(y^i,y^{[0]})),\dots \underset{y^{[1]}}{\operatorname{argmax}}(w^\top\phi(x^i,y^{[1]}) + \Delta_{p@1}(y^i,y^{[1]})
\end{aligned}
\tag{3.7}
$$

where we have made use of the definition of average precision from Eq. (3.1). Eq. (3.7) shows an important property: that the R most violating labelings can be found independently of each other using the precision loss at the required level of recall. This is the key property for the algorithm we propose in the following sub-section.

3.3.1 Inference and loss-augmented inference

Once the model is trained, testing it to report its AP requires, once again, the ability to produce a set of R predictions at the required levels of recall. Therefore, the key problems for both training and testing can be summed up, respectively, as:

$$\operatorname*{argmax}_{y^{[r]}}(w^\top \phi(x^i, y^{[r]}) + \Delta_{p@r}(y^i, y^{[r]})) \tag{3.8}$$

$$\operatorname*{argmax}_{y^{[r]}} w^\top \phi(x^i, y^{[r]}) \tag{3.9}$$

The algorithm we propose hereafter works interchangeably for both Eqs. (3.8) and (3.9), and also for the modified AP loss of Eq. (3.3). Given any ground-truth label sequence, y^i, the degrees of freedom of the precision loss are only the number of false positives, FP, and false negatives, FN. By making a prediction in left-to-right order along the sequence, the running values of FP and FN can only increment or remain unchanged. We can thus still approach the solution of Eq. (3.8) by dynamic programming, extending the state of a partial solution to include: a) the ground-truth label of the current frame, y_t, as in conventional Viterbi; b) the number of false positives, FP, in sub-sequence $y_{1:t}$; and c) the number of false negatives, FN, in sub-sequence $y_{1:t}$. We use notation $\psi(FP, FN, y_t)$ to indicate the $y_{1:t}$ sub-sequence with the highest score for the given extended state, and $s(\psi)$ for its score. The generic induction step is as follows: at any time step, t, a partial solution is obtained by extending two of the partial solutions of time $t-1$ with the current prediction, y_t, and correspondingly incrementing either FP or FN if the prediction is incorrect, or neither if correct. After the final time step, T, Eq. (3.8) is computed over the stored sequences and the argmax returned. Algorithm 3 describes the solution formally.

3.4 Experiments

The proposed approach has been evaluated on two challenging datasets of human activities, TUM Kitchen and CMU Multimodal Activity (CMU-MMAC). Descriptions and results for these two datasets are reported in the following sub-sections. The compared algorithms include: a) the proposed method based on the AP loss; b) structural SVM using the common 0-1 loss and Hamming loss, and c) a baseline offered by a standard SVM that classifies each frame separately. For SVM training, we have used constant $C = 0.1$ (based on a preliminary cross-validation), the RBF kernel (for non-linearity), and, for SSVM, convergence threshold $\varepsilon = 0.01$ (default). For the AP loss, given the greater computational complexity of the loss-augmented inference (approximately quadratic for sequences with sparse positives), we decode each sequence in sub-sequences of 300 frames each. To develop the software, we have used the SVM^{struct} package and its MATLAB wrapper [57, 138]. All experiments have been performed on a PC with an Intel i7 2.4GHz CPU with 8 GB RAM.

Algorithm 2 Algorithm for computing the loss-augmented inference of Eq. (3.8).

```
**Input**: $w$, $x = (x_1, \dots, x_T)$, $y^g = (y_1^g, \dots, y_T^g)$ (ground-truth labels), $r$
**Output**: $\bar{y}^{[r]}$
**Initialize:** $FP_{max} = FN_{max} = 0$

// FP, FN: running variables for the number of false positives and false negatives
// pos, neg: number of positives and negatives in $y^g$
// $\psi(invalidarg) = NULL, s(NULL) = -\infty$, [ ] = string concatenation operator

$\psi$ = FindHighestScoringSequences($w$, $x$, $y^g$);
$\bar{y}^{[r]}$ = FindMostViolatingLabeling($\psi$, $r$);
return $\bar{y}^{[r]}$
```

```
function FindHighestScoringSequences($w$, $x$, $y^g$)
// Finds all highest-scoring sequences for any combinations of FP and FN:

if $y_t^g = 0$
    $FP_{max} = FP_{max} + 1$
    for $FP = 0 : FP_{max}$, $FN = 0 : FN_{max}$, $t = 1 : T$
        $\psi(FP, FN, y_t = 0) =$
         $\text{argmax}(s([\psi(FP, FN, y_{t-1} = 0), 0]), s([\psi(FP, FN, y_{t-1} = 1), 0]))$
        $\psi(FP, FN, y_t = 1) =$
         $\text{argmax}(s([\psi(FP-1, FN, y_{t-1} = 0), 1]), s([\psi(FP-1, FN, y_{t-1} = 1), 1]))$
else
    $FN_{max} = FN_{max} + 1$
    for $FP = 0 : FP_{max}$, $FN = 0 : FN_{max}$, $t = 1 : T$
        $\psi(FP, FN, y_t = 0) =$
         $\text{argmax}(s([\psi(FP, FN-1, y_{t-1} = 0), 0]), s([\psi(FP, FN-1, y_{t-1} = 1), 0]))$
        $\psi(FP, FN, y_t = 1) =$
         $\text{argmax}(s([\psi(FP, FN, y_{t-1} = 0), 1]), s([\psi(FP, FN, y_{t-1} = 1), 1]))$
return $\psi$
end function
```

```
function FindMostViolatingLabeling($\psi$, $r$)
// Finds the labeling maximising the sum of score and loss:

$FN^* = round(pos\,(1-r))$ // sets the desired recall level

find $\text{argmax}_{\bar{y}^{[r]}} s(\bar{y}^{[r]})$ over $FP = 0 : neg$, $FN = FN^*$
  $\bar{y}^{[r]} = \text{argmax}_{\psi}$
  $[s(\psi(FP, FN, y_T = 0)) + \Delta_{p@r}(pos, FP, FN)$,
  $s(\psi(FP, FN, y_T = 1)) + \Delta_{p@r}(pos, FP, FN)]$

// for Eq. (3.9), just remove $\Delta_{p@r}$
// for the modified AP loss of Eq. (3.3), set $FN = 0 : FN^*$
```

Table 3.1 Comparison of the average precision over the TUM Kitchen dataset. SVM: standard SVM baseline; 0-1 loss and Hamming loss: structural SVM with conventional loss functions; AP loss: proposed technique.

	Average precision (%)			
Left hand sequences	SVM	0-1 loss	Hamming loss	AP loss
Reaching	24.5	44.8	18.5	**50.1**
TakingSomething	31.1	79.7	20.0	**80.7**
LoweringAnObject	19.3	44.6	16.9	**49.9**
ReleasingGrasp	18.1	53.2	25.0	**54.4**
OpeningADoor	10.9	9.1	9.1	**15.5**
ClosingADoor	9.2	9.1	9.1	**11.5**
OpeningADrawer	10.5	14.8	11.8	**20.6**
ClosingADrawer	10.9	9.1	9.1	**15.5**
Carrying	62.3	75.6	51.9	**80.2**
Mean	21.9	37.8	19.0	**42.0**
Right hand sequences	SVM	0-1 loss	Hamming loss	AP loss
Reaching	18.0	65.5	18.3	**68.9**
TakingSomething	12.8	**91.6**	14.1	90.9
LoweringAnObject	13.7	43.1	15.1	**47.7**
ReleasingGrasp	17.9	40.8	18.8	**45.4**
OpeningADoor	29.1	68.5	16.3	**73.9**
ClosingADoor	13.2	36.4	15.6	**41.3**
OpeningADrawer	14.7	26.8	13.8	**30.2**
ClosingADrawer	12.3	30.7	13.0	**38.0**
Carrying	58.7	85.4	63.1	**89.9**
Mean	21.3	54.3	20.8	**58.5**

3.4.1 Results on the TUM Kitchen dataset

The TUM Kitchen dataset is a collection of activity sequences recorded in a kitchen equipped with multiple sensors [130]. In the kitchen environment, various subjects were asked to set a table in different ways, performing 9 actions, *Reaching*, *TakingSomething*, *Carrying*, *LoweringAnObject*, *ReleasingGrasp*, *OpeningADoor*, *ClosingADoor*, *OpeningADrawer* and *ClosingADrawer*. For our experiments, we have chosen to use the motion capture (mocap) data from the left and right hands. These data consist of 19 sequences for each hand, each ranging in length between $1,000$ and $6,000$ measurements. Each measurement is a 45-D vector of 3D body joint locations.

Table 3.1 reports the results for activity recognition from the left and right hand sequences. The table shows that the mean of the AP over the nine classes is the highest for the proposed technique, with an improvement of 4.2 percentage points over the runner-up for both the left and right hand sequences. In addition, the proposed technique reports the highest average precision in all the classes with the left hand sequences, and in 8 classes out of 9 with the right hand sequences. In addition, the average precision of the proposed technique is about double that of the standard SVM baseline that does not leverage sequentiality.

Table 3.2 Comparison of the average precision over the CMU-MMAC dataset. SVM: standard SVM baseline; 0-1 loss and Hamming loss: structural SVM with conventional loss functions; AP loss: proposed technique.

	Average Precision (%)			
	SVM	0-1 loss	Hamming loss	AP loss
Close	9.9	**16.8**	9.0	16.2
Crack	11.4	23.1	8.3	**28.5**
None	30.9	46.6	29.8	**54.1**
Open	15.8	**33.3**	16.1	29.0
Pour	30.6	50.0	27.4	**61.4**
Put	13.1	27.8	11.4	**34.3**
Read	9.1	10.9	11.8	**16.5**
Spray	14.8	25.6	10.2	**28.4**
Stir	28.0	39.4	23.5	**45.5**
Switch-on	11.3	27.6	9.9	**32.9**
Take	22.5	47.1	19.8	**60.8**
Twist-off	10.1	25.5	7.9	**30.0**
Twist-on	9.8	19.0	8.1	**27.6**
Walk	10.4	23.2	9.6	**29.7**
Mean	16.3	29.7	14.5	**35.4**

3.4.2 Results on the CMU Multimodal Activity dataset

The CMU Multimodal Activity (CMU-MMAC) dataset contains multimodal measurements of the activities of 55 subjects preparing 5 different recipes: “brownies”, a salad, a pizza, a sandwich and scrambled eggs [24]. For our experiments, we have chosen to use the video clips of the 12 subjects preparing brownies from a dry mix box. The actions performed by the subjects are very realistic and are divided over 14 basic activities. The length of the 12 video clips ranges from $8,000$ to $20,000$ frames. For the experiments, we have used the first 8 videos for training and the remaining 4 for testing. For the feature vector of each frame, we have first extracted dense SIFT features at a 32-pixel step and used k-means with 32 clusters to generate a codebook. Then, the descriptors of each frame have been encoded into a $4,096$-D VLAD vector [139].

Table 3.2 reports the results for activity recognition over this dataset. The table shows that the mean of the AP is the highest for the proposed technique, with an improvement of 5.7 percentage points over the runner-up. In addition, the proposed technique reports the highest average precision for 12 classes out of 14, and more than doubles the SVM baseline.

3.5 Conclusion

The average precision has become a reference evaluation measure for its ability to assess performance at multiple operating points. However, the typical sequential labeling algorithms such as Viterbi do not allow the computation of the average precision. For this reason, in this chapter, we have proposed an inference procedure that infers a set of predictions at multiple levels of recall and allows measuring the average precision of a sequential classifier. In addition, we have proposed a structural SVM training algorithm for sequential labeling that minimises an average precision loss. Experiments conducted over two challenging activity datasets - TUM Kitchen and CMU-MMAC - have shown that the proposed approach significantly outperforms all of the other compared techniques and more than doubles the performance of a baseline. Moreover, while we have only focused on sequential labeling in this paper, the proposed approach could readily be employed for more general structures such as trees and graphs.

Chapter 4

Sequential Labeling With Structural SVM Under Non-decomposable Losses

In this chapter, we further the consideration of loss functions used for training structural SVM in the sequential labeling problem. We present a training algorithm for structural SVM that can minimise any other loss based on the classification contingency table, including the F_1 loss, precision/recall at fixed value of recall/precision, precision for a fixed value of positive predictions, precision/recall break-even point. Experimental results over a set of diverse and challenging datasets (TUM Kitchen, CMU-MMAC and Ozone Level Detection) are reported in terms of F_1 measure, detection rate and false alarm rate, showing that the proposed method outperforms other training approaches such as conventional SSVM and maximum likelihood.

4.1 Augmented inference for sequential labeling under loss functions of the classification contingency table

The solution of Eq. (2.78) is the crux of structural SVM for sequential labeling. Since the losses addressed in this paper do not decompose frame by frame, substantial changes to the conventional Viterbi algorithm are required. Here we consider all the loss functions that depend on the classification contingency table, i.e., the values of TP, FN, TN and FP. Given that the number of positives ($pos = TP + FN$) and negatives ($neg = TN + FP$) is known given a ground-truth sequence, y^g, the number of free parameters in the loss reduces to two. Correspondingly, we choose to parametrise loss $\Delta(y^g, y)$ using only FP and FN. As an example, we re-write the F_1 loss herewith as:

Algorithm 3 Algorithm for computing the loss-augmented inference (Eq. (2.78)) - part I.

```
Input: model $w$,
       sequence of measurements $x = (x_1, \dots, x_T)$,
       sequence of ground-truth labels $y^g = (y^g_1, \dots, y^g_T)$
Output: sequence of predicted labels $\bar{y} = (\bar{y}_1, \dots, \bar{y}_T)$
Initialize: $FP_{max} = FN_{max} = 0$

// Notations:
// FP, FN: running variables for the number of false positives and false negatives
// pos, neg: macros returning the number of positives and negatives in $y^g$
// $\psi(FP, FN, y_t)$: highest-scoring sequence up to frame t for given FP, FN and state $y_t$
// $s(\psi)$: score of sequence $\psi$ (uses arguments w and x)
// $\psi(invalidarg) = NULL, s(NULL) = -\infty$
// [ ]: string concatenation operator
// $\Delta \in \{F_1, p_{@r}, p_{@k}, p_{PRBEP}\}$: choice of loss function

$\psi$ = FindHighestScoringSequences($w$, $x$, $y^g$);
$\bar{y}$ = FindMostViolatingLabeling($\psi$, $\Delta$);
return $\bar{y}$

function FindHighestScoringSequences($w$, $x$, $y^g$)
// Finds all highest-scoring sequences for any combinations of FP and FN:

for $t = 1 : T$ do
  if $y^g_t = 0$ then $FP_{max} = FP_{max} + 1$
  else $FN_{max} = FN_{max} + 1$
  end if
  for $FP = 0 : FP_{max}$ do
    for $FN = 0 : FN_{max}$ do
      if $y^g_t = 0$ then
      $\psi(FP, FN, y_t = 0) = \text{argmax}(s([\psi(FP, FN, y_{t-1} = 0), 0]), s([\psi(FP, FN, y_{t-1} = 1), 0]))$
      $\psi(FP, FN, y_t = 1) = \text{argmax}(s([\psi(FP-1, FN, y_{t-1} = 0), 1]), s([\psi(FP-1, FN, y_{t-1} = 1), 1]))$
      else
      $\psi(FP, FN, y_t = 0) = \text{argmax}(s([\psi(FP, FN-1, y_{t-1} = 0), 0]), s([\psi(FP, FN-1, y_{t-1} = 1), 0]))$
      $\psi(FP, FN, y_t = 1) = \text{argmax}(s([\psi(FP, FN, y_{t-1} = 0), 1]), s([\psi(FP, FN, y_{t-1} = 1), 1]))$
      end if
    end for
  end for
end for

return $\psi$

end function
```

(continues)

Algorithm 4 Algorithm for computing the loss-augmented inference (Eq. (2.78)) - part II.

```
function FindMostViolatingLabeling($\psi$, $\Delta$)
// Finds the labeling maximising the sum of score and loss:
$best = -\infty$
1. if $\Delta == F_1$ then
for $FP = 0 : neg$ do
  for $FN = 0 : pos$ do
    $temp = \max(s(\psi(FP, FN, y_T = 0)), s(\psi(FP, FN, y_T = 1))) + \Delta_{F_1}(pos, FP, FN)$
    if $temp > best$ then
    $best = temp$
    $\bar{y} = \text{argmax}(s(\psi(FP, FN, y_T = 0)), s(\psi(FP, FN, y_T = 1)))$
    end if
  end for
end for
2. else if$\Delta == p_{@r}$ then
$FN^* = round(pos\,(1-r))$ // sets FN to desired recall level
  for $FP = 0 : neg$ do
    $temp = \max(s(\psi(FP, FN^*, y_T = 0)), s(\psi(FP, FN^*, y_T = 1))) + \Delta_{p_{@r}}(pos, FP, FN^*)$
    if $temp > best$ then
    $best = temp$
    $\bar{y} = \text{argmax}(s(\psi(FP, FN^*, y_T = 0)), s(\psi(FP, FN^*, y_T = 1)))$
    end if
  end for
3. else if$\Delta == p_{@k}$ then
for $FP = 0 : k$ do
  $FN^* = round(pos + FP - k)$ // sets FN to get desired k for given FP
  $temp = \max(s(\psi(FP, FN^*, y_T = 0)), s(\psi(FP, FN^*, y_T = 1))) + \Delta_{p_{@k}}(pos, FP, FN^*)$
  if $temp > best$ then
  $best = temp$
  $\bar{y} = \text{argmax}(s(\psi(FP, FN^*, y_T = 0)), s(\psi(FP, FN^*, y_T = 1)))$
  end if
end for
4. else if$\Delta == p_{PRBEP}$ then
for $FP = 0 : \min(pos, neg)$ do
  $FN^* = FP$ // sets FN equal to FP
  $temp = \max(s(\psi(FP, FN^*, y_T = 0)), s(\psi(FP, FN^*, y_T = 1))) + \Delta_{p_{PRBEP}}(pos, FP, FN^*)$
  if $temp > best$ then
  $best = temp$
  $\bar{y} = \text{argmax}(s(\psi(FP, FN^*, y_T = 0)), s(\psi(FP, FN^*, y_T = 1)))$
  end if
end for
end if
return $\bar{y}$
end function
```

$$\Delta_{F_1}(y^i, y) = 1 - \frac{2(pos - FN)}{2\,pos + FP - FN} \tag{4.1}$$

Our algorithm operates as follows: by making prediction y in left-to-right order along the sequence, the values of FP and FN can only increment or remain unchanged. Accordingly, we can still approach the solution of Eq. (2.78) by an extended Viterbi algorithm where the state of the solution up to frame t includes:

- the label of the state at time t, y_t, as in conventional Viterbi;
- the number of false positives, FP, in sequence $y_{1:t}$; and
- the number of false negatives, FN, in sequence $y_{1:t}$.

At any frame t, all the sequences built up to that frame ending with the same label and having the same values of FP and FN are equivalent in terms of the score that they can accumulate during the following frames. We therefore only need to retain the best of these equivalent sequences for future propagation. We use notation $\psi(FP, FN, y_t)$ to indicate the $y_{1:t}$ sequence with the highest score for the given extended state, and $s(\psi)$ for its score. The generic induction step is as follows: let us arbitrarily assume for convenience that the ground-truth label for frame t, y_t^g, is 0 and that we have determined the best sequences up to frame $t-1$. Let us now construct the best sequence up to frame t containing FP false positives, FN false negatives and ending in state $y_t = 1$. Such a sequence can only be:

- either the best sequence up to frame $t-1$ containing $FP-1$ false positives, FN false negatives and ending in state $y_{t-1} = 0$, appended with $y_t = 1$
- or the best sequence up to frame $t-1$ containing $FP-1$ false positives, FN false negatives and ending in state $y_{t-1} = 1$, appended with $y_t = 1$

since prediction $y_t = 1$ causes FP to increase by a unit and FN to remain unvaried. Analogous steps can be defined for the other three combinations of y_t^g and y_t. Algorithm 3 (part I) shows the detailed steps in pseudo-code, while a proof of correctness is provided in Appendix 1.

During the computation of Algorithm 3, the number of stored sequences increases at every time step and reaches its maximum at the last time step, T. Given that FP and FN are bound by the number of negative and positive labels, *neg* and *pos*, respectively, the number of stored sequences is upper bound by $2(neg+1)(pos+1)$.

The final stage of the algorithm searches over the best sequences for the sequence attaining the maximum of the sum of score and loss (Algorithm 3, part II). This loop can be adjusted to search for the maximum over any combinations of FP and FN. For instance:

- in the case of the F_1 loss, the loop searches over all values of FP and FN;
- in the case of the precision loss at r, it searches over all values of FP for a fixed value of FN;
- in the case of the precision loss at k, it searches over all values of FP for a corresponding, varying value of FN;
- in the case of the precision/recall break-even point loss, it searches over all equal values of FP and FN.

In general, the final stage can be easily adjusted to compute the loss-augmented inference for any loss that is a function of the contingency table.

4.2 Experiments

We have evaluated the proposed approach over a diverse set of datasets including two challenging datasets of human activities, TUM Kitchen and CMU Multimodal Activity (CMU-MMAC), and the Ozone Level Detection dataset. Descriptions and results for each of these datasets are reported in the following sub-sections. The compared algorithms include: a) the proposed method with the F_1 loss and AP loss as loss functions; b) structural SVM with the conventional zero-one loss and Hamming loss, and c) two baselines offered by a standard SVM that classifies each frame separately and an HMM trained with supervised maximum likelihood. The SVM provides a baseline for frame by frame classification which does not take sequentiality into account, and the HMM is a baseline for sequential models. For SVM training, we have used trade off constant $C = 0.1$, convergence threshold $\varepsilon = 0.01$, and RBF kernels. For the F_1 losses, given the greater computational complexity of the loss-augmented inference (approximately quadratic for sequences with sparse positives), we decode each sequence in sub-sequences of $B = 300$ frames each. For the HMM model, we have used Gaussian emission densities. To report performance, we use the detection rate (*DR*), the false alarm rate (*FAR*) and the F_1 measure. The detection and false alarm rates provide a breakdown of the accuracy over positive and negative samples, respectively, while the F_1 measure combines performance to allow for direct performance ranking. To develop the software, we have used MATLAB 2011 with the SVM^{struct} package from Thorsten Joachims and its

MATLAB wrapper from Andrea Vedaldi [57, 138]. All experiments have been performed on a PC with an Intel i7 2.4GHz CPU with 8 GB RAM.

4.2.1 TUM Kitchen dataset

The TUM Kitchen dataset is a collection of activity sequences recorded in a kitchen environment equipped with multiple sensors [130]. In this scenario, various subjects were asked to set the table using 9 types of basic activities, namely *Reaching*, *Carrying*, *TakingSomething*, *LoweringAnObject*, *ReleasingGrasp*, *OpeningADoor*, *ClosingADoor*, *OpeningADrawer*, *ClosingADrawer*. For our experiments, we have used the motion capture data (3-D body joint locations arranged in a 45-D vector) from the left and right hand, respectively. For each hand, there are 19 sequences ranging in length between $1,000$ and $6,000$ frames that we have split into a training set with 6 sequences and a test set with the remaining 13. For detection, we have trained a binary classifier per activity class.

Table 4.1 reports the results for the left hand, showing that training under the F_1 loss achieves the best F_1 measure, on average and for 7 actions out of 9. The average value is 40.3%, with the values varying significantly amongst the classes from a minimum of only 0.3% to a maximum of 72.1%. The reason for the lowest values is that some of the classes have a very small number of positive frames, and even moderate amounts of false positives result in the precision dropping drastically. Standard SVM reports a satisfactory average of 31.3%, but lower than that of SSVM with the same loss (Hamming). This shows that sequential classification can outperform an equivalent frame-by-frame classifier. The lowest average is achieved by HMM with only 4.3%. Table 4.2 reports the results for the right hand, showing an equivalent ranking amongst the classifiers (F_1 first and Hamming second).

To illustrate the behaviour of the different classifiers in a more immediate way, in Fig. 4.1 we show a result from action *OpeningADrawer*. In the sequence, the subject opens a drawer twice starting at frames 712 and 940, respectively. Fig. 4.1.a shows that SSVM with the Hamming loss, HMM and standard SVM do not detect either occurrence. SSVM with the 0-1 loss combines both occurrences into a single event, with major over-detection. Fig. 4.1.b shows that structural SVM with the F_1 loss manages to detect both occurrences rather accurately.

4.2.2 CMU Multimodal Activity dataset

The CMU Multimodal Activity (CMU-MMAC) dataset contains multimodal measurements of the activities of 55 subjects preparing 5 different recipes: “brownies”, salad, pizza, a sandwich and scrambled eggs [24]. For our experiments, we have chosen the video clips

Table 4.1 Comparison of F_1 measure, *DR* and *FAR* over the TUM Kitchen dataset (left hand sequences). F_1 loss, AP loss: proposed techniques; 0-1 loss and Hamming loss: structural SVM with conventional loss functions; HMM: hidden Markov model baseline; SVM: frame-by-frame SVM baseline

	Accuracy and F_1 measure(%)														
	0-1 loss			Hamming loss			F_1 loss			HMM			SVM		
	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	FAR
Reaching	9.6	*37.3*	82.5	29.2	*29.5*	7.6	28.3	*84.4*	51.4	8.1	*6.9*	8.2	**42.1**	*52.9*	12.7
TakingSomething	2.8	*11.1*	78.4	28.8	*81.1*	29.8	**29.2**	*93.0*	51.9	6.2	*5.4*	7.9	1.0	*0.5*	0.2
LoweringAnObject	4.5	*13.3*	59.7	**38.5**	*67.7*	17.7	34.0	*85.0*	39.9	7.6	*6.6*	8.3	35.7	*30.2*	4.9
ReleasingGrasp	12.2	*55.3*	82.3	21.4	*36.4*	11.8	**27.3**	*75.4*	41.4	5.6	*5.1*	8.4	7.8	*4.3*	0.6
OpeningADoor	0.3	*100*	86.8	0.2	*30.3*	32.2	**0.4**	*68.6*	46.7	0	*0*	0.1	0	*0*	0
ClosingADoor	0.3	*100*	83.4	0	*0*	20.2	**0.3**	*66.3*	49.7	0	*0*	0.1	0	*0*	0
OpeningADrawer	1.2	*33.5*	78.4	0	*0*	12.8	**3.0**	*59.8*	56.6	0	*0*	0	0	*0*	0
ClosingADrawer	1.2	*48.9*	90.1	1.1	*12.5*	23.5	**2.1**	*58.2*	58.2	0	*0*	0	0	*0*	0
Carrying	40.7	*37.1*	52.7	51.6	*69.5*	19.1	**72.1**	*74.6*	51.9	1.0	*0.5*	3.1	70.4	*85.6*	22.6
Average	8.8	*48.5*	77.1	33.5	*36.3*	19.4	**40.3**	*73.9*	49.7	4.3	*2.7*	4.0	31.3	*19.3*	4.6

Table 4.2 Comparison of F_1 measure, *DR* and *FAR* over the TUM Kitchen dataset (right hand sequences).

	Accuracy and F_1 measure(%)														
	0-1 loss			Hamming loss			F_1 loss			HMM			SVM		
	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	FAR
Reaching	7.5	*31.3*	77.4	**23.7**	*60.9*	38.2	12.9	*60.4*	41.6	0.9	*0.4*	3.4	2.5	*1.3*	0.1
TakingSomething	4.1	*16.7*	81.9	1.5	*3.8*	15.9	6.5	*15.0*	13.3	0.6	*0.3*	0.4	**20.5**	*13.0*	0.6
LoweringAnObject	5.3	*46.7*	86.7	5.3	*10.3*	14.9	12.1	*66.6*	50.4	0	*0*	0.4	**20.5**	*12.3*	0.4
ReleasingGrasp	14.0	*60.6*	73.6	13.1	*16.7*	14.4	**19.5**	*5.3*	46.8	0	*0*	0.4	2.9	*1.5*	0
OpeningADoor	1.4	*8.9*	79.6	**32.7**	*60.0*	13.7	22.2	*100*	46.4	0	*0*	0	7.3	*3.9*	0.2
ClosingADoor	4.0	*39.7*	86.6	8.8	*16.7*	12.4	16.5	*100*	47.9	0	*0*	0	**31.9**	*23.3*	1.1
OpeningADrawer	3.5	*25.9*	67.2	19.9	*58.9*	21.7	**15.7**	*100*	53.8	0	*0*	0	0	*0*	0
ClosingADrawer	5.3	*69.2*	89.5	11.5	*29.4*	11.8	**11.8**	*100*	54.5	0	*0*	0	0	*0*	0
Carrying	30.8	*30.7*	75.7	61.3	*54.2*	32.0	**72.8**	*69.2*	19.9	0	*0.0*	0	64.2	*93.0*	31.0
Average	9.1	*36.6*	79.8	27.4	*34.5*	19.4	**31.3**	*68.5*	43.5	0.1	*0.1*	0.5	21.3	*16.5*	3.7

Table 4.3 Comparison of F_1 measure, *DR* and *FAR* over the CMU-MMAC dataset.

	Accuracy and F_1 measure(%)														
	0-1 loss			Hamming loss			F_1 loss			HMM			SVM		
	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	FAR
Close	0.6	*100*	93.4	0.6	*45.5*	44.2	**1.5**	*100*	36.2	0	*0*	39.7	0	*0*	12.2
Crack	4.5	*87.0*	80.1	3.5	*26.6*	30.5	**11.6**	*88.8*	29.7	5.2	*25.9*	41.3	6.4	*30.7*	18.3
None	17.2	*13.3*	12.6	19.5	*16.8*	16.7	**29.9**	*39.7*	37.6	7.0	*10.1*	28.2	7.7	*5.4*	10.0
Open	8.4	*80.5*	94.1	12.4	*43.1*	30.0	**23.3**	*75.5*	25.6	8.5	*11.2*	28.6	9.6	*17.9*	14.5
Pour	12.6	*13.3*	17.6	**26.6**	*75.4*	70.2	19.6	*29.4*	29.8	11.2	*18.6*	37.6	13.6	*17.5*	25.1
Put	9.8	*55.8*	50.5	7.9	*47.6*	54.7	**24.7**	*83.0*	25.1	9.1	*27.9*	29.2	8.8	*21.6*	18.9
Read	3.6	*100*	67.8	0.8	*19.4*	62.5	**8.1**	*98.2*	27.9	0	*0*	19.4	0	*0*	8.3
Spray	2.2	*100*	98.9	1.3	*3.5*	10.2	**17.3**	*100*	23.0	2.9	*14.6*	43.1	1.5	*8.7*	26.0
Stir	33.4	*51.0*	61.5	14.8	*10.1*	10.2	**43.2**	*77.4*	71.8	31.8	*70.6*	63.4	34.2	*71.5*	74.4
Switch-on	6.4	*97.4*	95.6	4.3	*24.1*	33.1	**8.9**	*69.8*	46.7	4.5	*20.4*	34.8	3.0	*13.4*	26.4
Take	10.9	*15.3*	24.4	19.5	*39.0*	35.1	**48.4**	*82.9*	21.4	7.0	*11.0*	29.8	7.5	*11.6*	26.2
Twist-off	1.5	*93.8*	95.2	2.4	*77.5*	47.6	**3.5**	*77.7*	32.5	0	*0*	28.4	0	*0*	20.2
Twist-on	1.1	*82.5*	97.9	2.0	*46.4*	29.3	**6.8**	*100*	18.0	0	*0*	33.6	0	*0*	18.3
Walk	1.7	*100*	93.4	0	*0*	9.6	**5.2**	*87.3*	27.0	0	*0*	37.5	0.4	*5.6*	20.8
Average	7.0	*70.7*	70.0	8.7	*33.9*	34.6	**22.1**	*79.3*	32.3	10.0	*15.0*	35.3	10.3	*14.6*	22.8

Table 4.4 Comparison of F_1 measure, *DR* and *FAR* over the Ozone Level Detection dataset.

	Accuracy and F_1 measure(%)														
	0-1 loss			Hamming loss			F_1 loss			HMM			SVM		
	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	*FAR*	F_1 measure	*DR*	FAR
Eighthr	6.5	*79.3*	94.3	0	*0*	0.2	**21.3**	*34.5*	7.8	8.7	*51.7*	43.1	4.9	*3.5*	1.6
Onehr	2.3	*80*	95.6	0	*0*	0	**9.5**	*10*	1.4	2.1	*60*	78.3	0	*0*	2.8

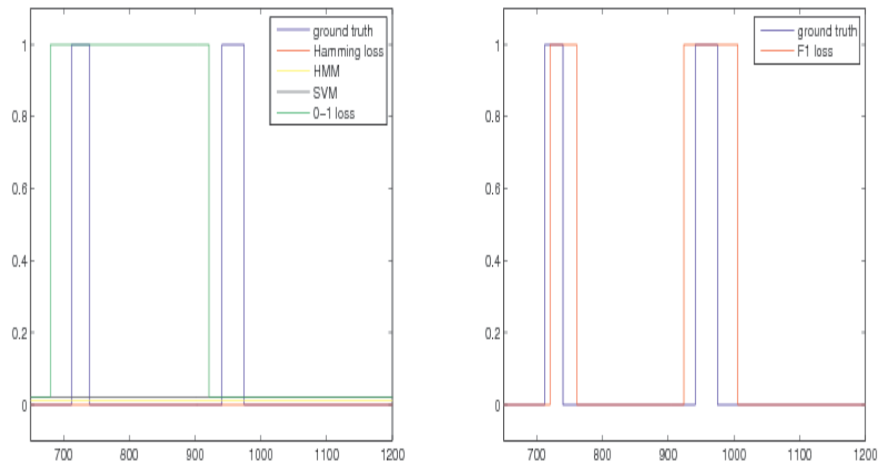


Fig. 4.1 An example of results from TUM Kitchen sequence 16, left hand, for action *OpeningADrawer*: a) structural SVM with the 0-1 loss and Hamming losses, HMM and standard SVM versus the ground truth; b) structural SVM with the F_1 loss versus the ground truth. This figure is better viewed in colour.

of the 12 subjects preparing brownies from a dry mix box. The subjects were given the recipe, but they were left to prepare the food in a spontaneous way without receiving any instructions on how to perform the task. Therefore, the sequence of actions and the overall video length vary greatly across subjects, making the dataset very realistic. The actions performed by the subjects are divided into 14 classes including *Pouring*, *Stirring*, *Reading* and others. As the video source, we have used static camera *7151062* which offers a side view of the scene. As ground truth, we have utilised the only available annotation which was provided for the head-mounted camera. Given that the side and head views differ, the annotation is moderately inaccurate and may be by itself a source of error. The length of these 12 videos ranges from $8,000$ to $20,000$ frames. As split, we have used the first 8 videos for training and the remaining 4 for testing. To form a feature vector for each frame, we have first extracted dense SIFT features at a 32-pixel step and used k-means with 32 clusters to generate the codebook. Then, the descriptors of each frame have been encoded into a $4,096$-D Vector of Linearly Aggregated Descriptors (VLAD) [139].

Table 4.3 reports the results, showing that the best average F_1 measure is achieved, again, by training under the F_1 loss. The lower value of 22.1% compared to the TUM Kitchen dataset shows that CMU-MMAC is a more probing scenario. Training under the F_1 loss also achieves the best F_1 measure for 13 classes out of 14. The second best average is achieved, again, by the baselines, SVM and HMM, with 10.3% and 10.0%, respectively.

4.2.3 Ozone Level Detection dataset

The Ozone Level Detection dataset is a time series containing daily atmospheric measurements (collected from 1 January 1998 to 31 December 2004 at Houston, Galveston and the Brazoria area) and corresponding binary annotations i.e. “ozone/non-ozone day”. The time series contains a total of $2,536$ frames of 72-D dimensionality. There are two distinct annotations, one from direct ozone measurements at eight-hour intervals (*eight-hour*) and the other at one-hour intervals (*one-hour*). Given that some of the measurements are missing, we have retained only the frames with complete dimensions, reducing the total data to a training set of $1,629$ frames and a test set of 730.

Table 4.4 reports the results for both annotations, showing once more that training under the F_1 loss obtains the highest F_1 measure thanks to the best trade off between positive and negative samples. The second best is the zero-one loss and HMM. The ranking of the classifiers is similar to those over the activity recognition datasets, showing that the performance is stable over data of varying kinds.

4.3 Conclusion

In this chapter, we have proposed algorithms for training structural SVM for sequential labeling under non-decomposable loss functions. In contrast to the more conventional zero-one and Hamming losses, such loss functions are more suited when specific levels of precision and recall are sought and are particularly useful in detection scenarios where the number of positive samples is significantly lower than that of negative samples. The loss functions covered include the precision at k, the precision at a given level of recall, the precision/recall break-even point, the F_1 loss and any other loss that is a function of the classification contingency table. Experiments conducted over sequential datasets for detection including the challenging TUM Kitchen and CMU-MMAC activity datasets and the Ozone Level Detection dataset have shown that the proposed technique systematically outperforms other techniques in terms of the F_1 measure and achieves the most interesting performance trade-offs between positive and negative samples. While this paper focuses on the sequential case, the proposed technique is more general and can be employed with other structures such as trees and forests.

Chapter 5

Structural SVM With Partial Ranking for Activity Segmentation and Classification

In this chapter, we consider the problem of training structural SVM under multiple ground-truth labelings with different levels of correctness. While the margin constraint guarantees that the ground truth labeling receives a higher score than any other labelings, it does not ensure that the other labelings are ranked in correctness order. In order to ensure that labelings which are close to the ground truth receive a higher score than other, less qualified labelings, we propose partial-ranking structural SVM (PR-SSVM) and test the proposed approach over two challenging activity datasets: the TUM Kitchen dataset and the CMU-MMAC dataset. The experimental results show that the proposed method achieves an accuracy level higher than that of conventional structural SVM and one that is remarkably higher than previous results.

5.1 Introduction and related work

Structured prediction addresses the joint assignment of a set of class labels from a set of measurements in the presence of dependencies between the labels. This is a frequent situation with examples ranging from classification of web pages, prediction of protein structure, and natural language parsing to segmentation and classification of human activities[4, 93, 123]. Compared to the separate assignment of single labels, the structured approach is expected to be more accurate by leveraging the relationships among the labels. The structure is commonly represented in terms of a graphical model, and training and inference algorithms are employed to provide the parametrization of the model and label prediction.

Among the possible training approaches, structural SVM was proposed to extend the large-margin concept of the support vector machine (SVM) to the structured case [128, 134]. It has been applied to a variety of structured tasks with remarkable experimental accuracy [148, 3, 170, 58, 28]. Training of structural SVM is performed by imposing a pre-determined margin between the score granted to the ground-truth labels and the score granted to any other labeling. Since a predicted labeling may differ from the ground truth to a different extent (from almost correct to completely incorrect), a graded margin such as the Hamming distance is often used. However, while the margin constraint guarantees that the ground-truth labeling receives a higher score than all other labelings, it does not ensure that the other labelings are ranked in correctness order. This may affect applications such as, for instance, human activity segmentation where the manual annotation of the start and end of an activity has a significant degree of uncertainty. In this case, we may wish to ensure that also labelings which are close to the ground truth receive a score higher than other, less qualified labelings. Therefore, the idea proposed in this paper is to augment the constrained optimization of structural SVM with an additional set of constraints ensuring proper scoring of additional, selected labelings. To this aim, we define a modified Hamming loss to measure the distance between an arbitrary labeling and a predicted labeling. We refer to the proposed technique as partial-ranking structural SVM (PR-SSVM) hereafter.

The task tackled in this paper is the joint segmentation and classification of human activities. In formal terms, we aim to optimally infer a sequence of class labels, $y = \{y_1, \dots, y_t, \dots, y_T\} \in Y$, from a given sequence of measurements, $x = \{x_1, \dots, x_T\}$. We perform classification by detection where $y_t \in \{0, 1\}$, $y_t = 1$ means the presence of an assigned action, and $y_t = 0$ its absence. Following a common model, we assume that the labels are connected in a first-order Markov chain and that each label is connected to the measurement with the same time index. Optimal inference for this model is efficiently provided by dynamic programming algorithms while training is performed by the method described in the following sections. The proposed approach has been tested over two challenging activity sequence datasets: the TUM Kitchen mocap dataset [130] and the CMU-Multimodal Activity video dataset (CMU-MMAC) [24]. The experimental results show that the proposed method achieves an accuracy higher than that of conventional structural SVM and it is also remarkably higher than previous results.

5.2 Loss Function and Partial Ranking Training

5.2.1 Loss function

In structural SVM, the margin imposed between the ground-truth labeling, y^g, and a predicted labeling, y, varies according to a chosen loss function, $\Delta(y^g, y)$, which quantifies the loss carried by a misprediction. The choice of loss function is typically restricted to functions that decompose over the single labels of a labeling since this facilitates efficient training. The most common choice is the Hamming loss:

$$\Delta_H(y^g, y) = \frac{1}{T}\sum_{t=1}^{T} \delta(y_t^g \neq y_t) \tag{5.1}$$

where $\delta(true) = 1, \delta(false) = 0$ and $y_t^g, y_t, t = 1 \dots T$, are the individual labels in labelings y^g and y, respectively.

In order to augment structural SVM with an additional set of constraints, a loss function is needed between a reference labeling (other than the ground truth) and a prediction. We note such a labeling as $\tilde{y}^g$ and the new loss function as $\Delta'(y^g, \tilde{y}^g, y)$. A natural way to define it is as difference of losses with respect to the ground truth:

$$\begin{aligned} \Delta'(y^g, \tilde{y}^g, y) &= \Delta_H(y^g, y) - \Delta_H(y^g, \tilde{y}^g) \\ &= \frac{1}{T}\sum_{t=1}^{T} \left(\delta(y_t^g \neq y_t) - \delta(y_t^g \neq \tilde{y}_t^g)\right) \end{aligned} \tag{5.2}$$

In this way, also this loss function remains decomposable over single labels and retains efficient training. Its minimum is a negative value occurring at $\Delta'(y^g, \tilde{y}^g, y = y^g)$, that is, when the prediction is equal to the ground truth. In fact, function $\Delta'(y^g, \tilde{y}^g, y)$ as defined in (5.2) is a hybrid loss/gain function still rewarding similarity to the ground truth.

5.2.2 Training by partial ranking

Given a loss function and a ground-truth labeling, all labelings can be ranked in loss order to form a totally ordered set. In principle, any scoring classifier can be trained not only to assign the highest score to the ground truth, but also to score all labelings in loss order. However, in the structured case the number of distinct labelings is exponential and such an approach would prove infeasible. Therefore, in this work we propose to impose only a partial order relation amongst the labelings by selecting a sub-set to be scored in loss order. We refer to this approach as *partial ranking* for short. While the sub-set can be chosen in any arbitrary way, we argue that selecting labelings which are small perturbations of the ground truth may improve the classifier's accuracy, especially in cases where the ground truth has a degree of uncertainty. In this work, we deal with sequences of binary labels and choose to add only one labeling per sample, $\tilde{y}^g$, obtained by modifying the ground truth by setting to 1 any 0 labels preceding and following the ground truth' 1 labels:

$$\tilde{y}^g_t = 1 \;\text{ if } y^g_t = 1 \;\text{ or } y^g_{t-1} = 1 \;\text{ or } y^g_{t+1} = 1 \tag{5.3}$$

so as to accommodate annotation uncertainty about both the start and the end of a run of positive samples.

For more general cases with multi-valued labels or non-sequential structures, one can build ground-truth perturbations either randomly or manually. For instance, individual labels can be set to the values that are semantically most similar to the ground truth (such as "orange" for "red" or "adverb" for "adjective"): in general, labelings with small loss with respect to the given ground truth.

5.3 Extended Primal Problem

The extension provided by PR-SSVM to the objective function of Eq. (2.77) consists of the introduction of additional constraints ensuring the score ranking of labelings other than the ground truth. The extended problem is expressed as:

$$\begin{aligned}
&\underset{w,\xi,\tilde{\xi}}{\operatorname{argmin}} \|w\|^2 + C\sum_{i=1}^{N}(\xi^i + \tilde{\xi}^i) \quad s.t.\\
&w^T\phi(x^i,y^i) - w^T\phi(x^i,y) \geq \Delta(y^i,y) - \xi^i,\\
&w^T\phi(x^i,\tilde{y}^i) - w^T\phi(x^i,y) \geq \Delta'(y^i,\tilde{y}^i,y) - \tilde{\xi}^i,\\
&\quad i = 1\ldots N,\; \forall y \in Y
\end{aligned} \tag{5.4}$$

Adding the new constraints brings their total number to $2N\,|Y|$. However, the working-set approach still applies and the loss-augmented inference becomes:

$$\begin{aligned}
\tilde{y}^{*i} &= \underset{y}{\operatorname{argmax}}(w^T\phi(x^i,y) + \Delta'(y^i,\tilde{y}^i,y))\\
&= \underset{y}{\operatorname{argmax}}(w^T\phi(x^i,y) + \Delta(y^i,y) - \Delta(y^i,\tilde{y}^i))\\
&= \underset{y}{\operatorname{argmax}}(w^T\phi(x^i,y) + \Delta(y^i,y))
\end{aligned} \tag{5.5}$$

One can see that Eq. (5.5) is formally identical to Eq. (2.78) and returns the same labeling, i.e. $\tilde{y}^{*i} \equiv y^{*i}$. However, variable $\tilde{\xi}^i$ is set by the different loss:

$$\tilde{\xi}^i = -w^T \phi(x^i, \tilde{y}^i) + w^T \phi(x^i, y^{*i}) + \Delta'(y^i, \tilde{y}^i, y^{*i}) \quad (5.6)$$

The combination of both ξ and $\tilde{\xi}$ in the objective eventually leads to the extended solution.

5.4 Experimental Results and Discussion

In this section, we evaluate the proposed method on two challenging human activity datasets: the TUM Kitchen dataset and the CMU Multimodal Activity (CMU-MMAC) dataset [130, 24]. The TUM Kitchen dataset is a collection of activity sequences recorded in a kitchen equipped with multiple sensors [130]. Four human subjects were asked to set a table using 9 actions, namely *Reaching*, *Carrying*, *TakingSomething*, *LoweringAnObject*, *ReleasingGrasp*, *OpeningADoor*, *ClosingADoor*, *OpeningADrawer*, *ClosingADrawer*. For our experiments, we have used the data from the motion capture sensor for the right and left hands which encode the relevant 3D joints as a 45-D vector. The total number of sequences is 19, each ranging in length between $1,000$ and $6,000$ frames. The CMU-MMAC dataset contains activity sequences from 55 subjects preparing food from various recipes [24]. For our experiments, we have selected the 12 subjects preparing brownies from a dry-mix box, with the activities labeled in 14 classes (see Table 5.4 for the complete list). The videos are from side-view camera *7151062*, with a duration ranging between $8,000$ and $20,000$ frames each.

For performance comparison, we have selected the following classifiers: a) as baseline, a standard support vector machine assigning each frame to an activity (*Baseline*); b) a structural SVM classifier using the conventional constraints over the ground-truth labelings (*SSVM*); c) the proposed technique with the augmented set of constraints (*PR-SSVM*). All classifiers were implemented in detector style as a set of binary classifiers, one per activity class. For evaluation, we have recorded performance in terms of detection rate (*DR*), false alarm rate (*FAR*) and $F1$ score. While the detection and false alarm rates describe the trade-off between sensitivity and robustness, the $F1$ score summarizes the performance in a single figure. As parameters, we have set $C = 0.1$ and $\varepsilon = 0.01$ and used a linear kernel for all classifiers. The software for PR-SSVM ans SSVM was developed using the SVM^{struct} package and its MATLAB wrapper [57, 138], while *libsvm* was used for the baseline [15].

Table 5.1 Comparison of detection rate, false alarm rate and $F1$ score on the TUM Kitchen dataset (right hand).

	DR (%)			*FAR* (%)			$F1$ score (%)		
Activity	Baseline	SSVM	PR-SSVM	Baseline	SSVM	PR-SSVM	Baseline	SSVM	PR-SSVM
Reaching	0	60.9	40.2	0	38.2	24.6	0	**23.7**	21.9
TakingSomething	13.2	3.8	10.3	0.3	15.9	10.3	**22.2**	1.5	5.4
LoweringAnObject	0	10.3	27.8	0	14.9	15.1	0	5.3	**13.6**
ReleasingGrasp	0	16.7	18.0	0	14.4	13.4	0	13.1	**17.9**
OpeningADoor	33.9	60.0	64.3	0.6	13.7	6.4	47.4	32.7	**49.2**
ClosingADoor	0	16.7	48.9	0	12.4	5.4	0	8.8	**37.2**
OpeningADrawer	0	58.9	67.1	0	21.7	13.1	0	19.9	**31.3**
ClosingADrawer	0	29.4	17.5	0	11.8	8.4	0	**11.5**	10.1
Carrying	98.0	54.2	57.9	38.7	32.0	24.7	**84.2**	61.3	64.3
Average	16.1	45.2	48.0	4.4	18.6	23.7	21.3	27.4	**39.0**

5.4.1 Results on the TUM Kitchen dataset

For TUM Kitchen, we have split the data into a training set with 6 sequences (namely, episodes 1-1 to 1-4, 0-2 and 0-12) and a test set with the remaining 13. Tables 5.1 and 5.2 report the accuracy results at frame level for each class and as average over the classes. These tables show that the tested classifiers achieve very different trade-offs between detection and false alarm rates:

- the baseline reports the lowest *DR* and *FAR*, implying that most activities will simply go undetected and that its training is biased by the most frequent class (negative);
- structural SVM has a much higher detection rate than the baseline, yet with a rather high *FAR* (on average, 18.6% for the right hand and 16.8% for the left);
- the proposed technique, PR-SSVM, achieves the best trade-off as it obtains a higher *DR* than SSVM (on average, 48.0% vs. 45.2% for the right hand, and 37.1% vs. 36.3% for the left hand), together with a lower *FAR* (12.7% vs. 18.6% for the right hand, and 9.7% vs. 16.8% for the left hand).

Since it is difficult to rank classifiers based on two rates, we use the $F1$ score for direct comparison. Tables 5.1 and 5.2 show that PR-SSVM reports the highest $F1$ score on average and for 5 classes out of 9 for the right hand, and on average and for 7 classes out of 9 for the left one. The average improvement ranges from 11.6 to 19.4 percentage points over SSVM and from 17.8 to 21.8 percentage points over the baseline. Fig. 5.1 shows a typical behavior where a) the baseline misses the activity altogether, (b) SSVM over-segments the activity, while (c) PR-SSVM detects the entire activity as a single segment. Eventually, Table 5.3 compares the frame-level accuracy with previous results: although these results cannot be compared directly as the training and test sets differ, the proposed technique shows a remarkable improvement of over 14 percentage points over the closest result.

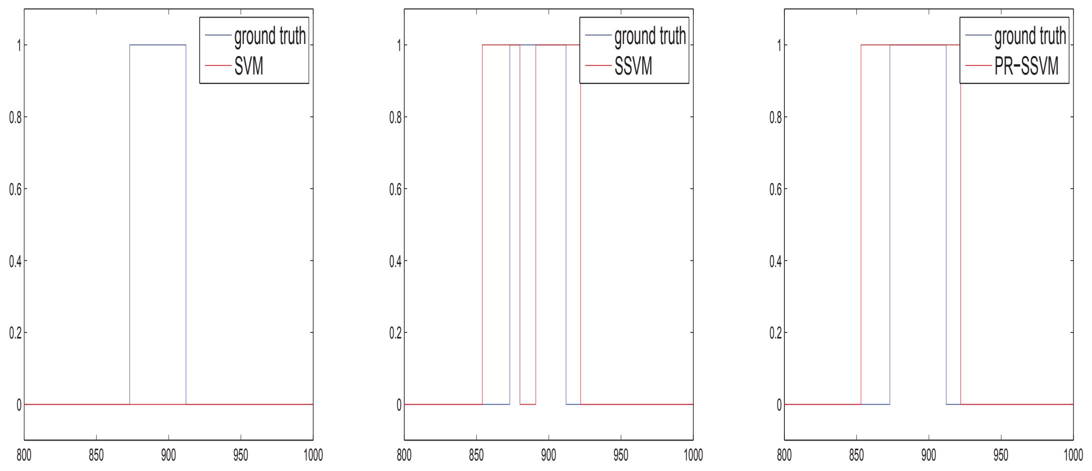


Fig. 5.1 Example of detection (video 21, right hand, action *ClosingADoor*): a) Baseline; b) Structural SVM (SSVM); c) Partial-ranking structural SVM (PR-SSVM).

Table 5.2 Comparison of detection rate, false alarm rate and $F1$ score on the TUM Kitchen dataset (left hand).

	DR (%)			*FAR* (%)			$F1$ score (%)		
Activity	Baseline	SSVM	PR-SSVM	Baseline	SSVM	PR-SSVM	Baseline	SSVM	PR-SSVM
Reaching	0.2	29.5	28.9	0.0	7.6	5.9	0.4	29.2	**33.4**
TakingSomething	50.2	81.1	68.7	0.7	29.8	10.7	9.2	28.8	**52.8**
LoweringAnObject	53.5	67.7	78.0	6.3	17.7	15.9	**52.6**	38.5	51.3
ReleasingGrasp	0	36.4	22.5	0	11.8	5.2	0	21.4	**26.5**
OpeningADoor	0	30.3	0	0	32.2	14.1	0	**0.2**	0
ClosingADoor	0	0	18.5	0	20.2	16.2	0	0	**2.4**
OpeningADrawer	0	0	7.8	0	12.8	2.1	0	0	**6.2**
ClosingADrawer	0	12.5	37.3	0	23.5	4.9	0	1.1	**17.0**
Carrying	90.5	69.5	72.1	26.1	19.1	14.1	**85.0**	51.6	78.1
Average	21.8	36.3	37.1	3.7	16.8	9.7	23.7	33.5	**52.9**

Table 5.3 Comparison of frame-level accuracy with previous results for the TUM Kitchen dataset.

Method	Average accuracy
CRF [130]	62.8
Switching model [5]	70.1
Proposed method	85.0

5.4.2 Results on the CMU-MMAC dataset

For CMU-MMAC, we have divided the 12 sequences into a training set with the first eight and a test set with the remaining four. As features, we have extracted a dense set of SIFT features from each frame and encoded them as a vector of linearly aggregated descriptors (VLAD) using k-means with 32 clusters [51]. Each measurement results in a $4,096$-D vector.

Table 5.4 Comparison of detection rate, false alarm rate and $F1$ score on the CMU-MMAC dataset (“brownies”).

	DR (%)			*FAR* (%)			$F1$ score (%)		
Activity	Baseline	SSVM	PR-SSVM	Baseline	SSVM	PR-SSVM	Baseline	SSVM	PR-SSVM
Closing	0	45.5	54.5	12.5	44.2	40.0	0	0.6	**1.0**
Cracking	32.5	26.6	73.4	18.2	30.5	31.2	**6.5**	3.5	5.3
None	5.8	16.8	28.4	10.3	16.7	29.4	7.5	19.5	**25.1**
Opening	16.9	43.1	45.1	13.3	30.0	33.7	9.8	12.4	**15.2**
Pouring	18.8	75.4	70.2	25.0	70.2	59.6	14.0	26.6	**27.6**
Putting	25.4	47.6	52.4	19.2	54.7	33.2	8.7	7.9	**12.1**
Reading	0	19.4	15.6	8.0	62.5	53.1	0	0.8	**3.7**
Spraying	7.8	3.5	14.9	24.1	10.2	22.0	1.6	1.3	**1.6**
Stirring	71.5	10.1	29.8	70.2	10.2	29.1	**35.2**	14.8	29.4
Switching on	15.6	24.1	44.6	25.8	33.1	23.6	3.8	**4.3**	4.2
Taking	21.6	39.0	15.3	28.7	35.1	14.1	7.5	**19.5**	13.9
Twisting off	0	77.5	65.9	19.5	47.6	40.0	0	2.4	**3.6**
Twisting on	0	46.4	49.8	17.3	29.3	20.2	0	2.0	**2.5**
Walking	8.8	0	19.5	18.4	9.6	21.0	0.5	0	**8.3**
Average	16.1	30.0	40.5	22.2	41.2	26.9	7.4	8.7	**19.6**

Table 5.4 reports the accuracy results, showing that the relative ranking of the classifiers is unvaried:

- the baseline reports, again, both the lowest *DR* and *FAR*, with a *DR* of only 14.6% on average;
- the proposed technique, PR-SSVM, achieves both a higher average *DR* (40.5% vs. 30.0%) and lower *FAR* (26.9% vs. 41.2%) than standard structural SVM;
- PR-SSVM reports the highest $F1$ score on average and for 10 classes out of 14, with an average improvement of 10.9 points over SSVM and 9.3 points over the baseline.

Table 5.5 Comparison of frame-level accuracy with previous results for the CMU-MMAC dataset.

Method	Average accuracy
HMM-MIO [167]	38.4
CRF [172]	38.8
Proposed method	69.8

Again, Table 5.5 compares the frame-level accuracy with existing results, showing a remarkable improvement of over 31 percentage points over the closest value.

5.5 Conclusion

In this letter, we have proposed a novel technique for structured prediction enforcing a partial ranking among predicted labelings. This technique is an extension of the versatile structural SVM which joins maximum-margin training with the ability to predict co-dependent labels.

The proposed technique, named partial ranking structural SVM (PR-SSVM), imposes a score margin between additional labelings other than the ground truth. In particular, in this paper we have enforced a margin between "almost-correct" labelings and the remaining labelings for sequential classification of activities. The results over two contemporary and challenging datasets (TUM Kitchen and CMU-MMAC) show that:

- compared with a baseline classifier providing single-frame classification and standard structural SVM, the proposed PR-SSVM always achieves the highest average $F1$ scores, with improvements ranging between 9 and 11 percentage points (Tables 5.1, 5.2 and 5.4);
- compared with the other two classifiers, PR-SSVM achieves the most appealing trade-off between *DR* and *FAR*, with averages always above standard structural SVM;
- compared with previous results, PR-SSVM obtains an improvement of over 14 percentage points on TUM Kitchen and 31 points on CMU-MMAC (Tables 5.3 and 5.5).

In addition, the proposed partial ranking extension is not restricted to sequential classification, but can be applied to any label structure and any sub-set of constraints. Since the proposed loss decomposes over single labels, the efficient loss-augmented inference proper of structural SVM is retained.

Chapter 6

Conclusion

This PhD thesis mainly deals with sequential labeling models for human activity segmentation and classification. While there already exists significant research work on this topic, including the hidden Markov model (HMM), conditional random fields and structural SVM, it is still an ongoing research issue.

We have first reviewed the literature and related work in Chapter 2 including human activity taxonomies, joint activity segmentation and classification, sequential labeling, HMM, SVM, multi-label classification and so on. Training an HMM by likelihood maximization does not always achieve satisfactory run-time accuracy. SVM, on the other hand, is trained by minimising the empirical loss over the set of examples instead of the likelihood function and often leads to better performance. Conventional SVM treats the training instances as independent and identically distributed (i.i.d.) which just ignores the sequential feature between instances. Furthermore, the original formulation of the SVM uses only the 0-1 loss as loss function.

Joachims in [56] had proposed an algorithm for training SVM under different loss functions. However, they didn't consider the case of sequential labeling. Tsochantaridis *et al.* have hinted at the possibility of training SVM for sequential labeling under various loss functions but didn't give explicit algorithms.

For this reason, and considering that loss functions available for training structural SVM are restricted to decomposable cases such as the 0-1 loss and the Hamming loss, we have presented an approach for training SVM for sequential labeling under the F_1 loss, the precision (recall) at a set point of recall (precision), the precision for a fixed number of positive predictions, the precision/recall break-even point and any non decomposable loss based on the classification contingency table in Chapter 3. Such losses are more appropriate than the decomposable cases for many real-world problems, especially when specific levels of precision and recall are sought and applied to particular useful in detection scenarios where the number of positive samples is significantly lower than that of negative samples. In addition, we have presented an algorithm for training SSVM under a formulation of the average precision loss, and we have also proposed an algorithm for evaluating the average precision of the sequential inference in Chapter 4.

Furthermore, we have proposed a novel technique for structured prediction with a "partial ranking" among predicted labelings which we have called PR-SSVM. We ensure that the score of the "almost correct" ground truth labeling is lower than that of the ground truth labeling, but higher than any other remaining labeling for sequential classification of activities in Chapter 5. This is done in order to modify the uncertainty of the ground truth.

All experimental results are conducted over three challenging sequential datasets, namely the TUM Kitchen dataset, the CMU-MMAC activity dataset and the Ozone Level Detection dataset. We choose multiple performance metrics (the F_1 measure, the Detection Rate and the False Alarm Rate) to show that our proposed approaches outperform the state-of-the-art methods for structural SVM and lead to significant improvement. For example, our proposed approach of non decomposable loss functions always achieves the highest value of F_1 measure on average and for most actions out of all, leading to an improvement of up to 13.4% of the F_1 measure compared to that of the conventional structural SVM and also 4.3% improvement of classification for the left hand sequences of the TUM Kitchen dataset. For the approach of training algorithm that maximizes an approximation of the average precision, it achieves up to 2.6% on the TUM Kitchen dataset and 6.6% on the CMU-MMAC dataset of the F_1 measure. For the PRSSVM, it achieves improvements ranging between 9% to 11% of F_1 measure. And compared with previous results, PRSSVM obtains an improvement of over 14% on the TUM Kitchen dataset and 31% on the CMU-MMAC dataset. Besides, they all achieve the best performance trade-offs between the Detection Rate and False Alarm Rate.

While this thesis has focused on the sequential case, the proposed techniques are more general and can straightforwardly be employed with other structures such as trees and forests. On the other hand, our proposed partial ranking extension is not restricted to sequential classification, but can be applied to any label structure and any sub-set of constraints. Since the proposed loss is decomposable over single labels, the efficient loss-augmented inference of structural SVM is retained. Another interesting objective could be to extend the proposed techniques to the increasingly growing area of deep learning, in particular to recurrent neural networks such as the Elman and Jordan networks and the long short-term memory [84, 68]. These are our future goals, and we will keep researching them.

References

[1] Ahad, A. R., Ogata, T., Tan, J. K., Kim, H., and Ishikawa, S. (2008). Motion recognition approach to solve overwriting in complex actions. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–6. IEEE.

[2] Ali, S. and Shah, M. (2010). Human action recognition in videos using kinematic features and multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):288–303.

[3] A.Zien, U. and T.Scheffer (2007). Transductive support vector machines for structured variables. In *24th Tnternational Conference on Machine Learning*, New York, USA. IEEE.

[4] Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. (2007). *Predicting Structured Data*. The MIT Press.

[5] Bargi, A., Da Xu, R. Y., and Piccardi, M. (2012). An online hdp-hmm for joint action segmentation and classification in motion capture data. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7. IEEE.

[6] Batra, D., Chen, T., and Sukthankar, R. (2008). Space-time shapelets for action recognition. In *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*, pages 1–6. IEEE.

[7] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer.

[8] Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, volume 4. springer New York.

[9] Blackburn, J. and Ribeiro, E. (2007). Human motion recognition using isomap and dynamic time warping. In *Human Motion–Understanding, Modeling, Capture and Animation*, pages 285–298. Springer.

[10] Blank, M., Gorelick, L., Shechtman, E., Irani, M., and Basri, R. (2005). Actions as space-time shapes. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1395–1402. IEEE.

[11] Bobick, A. (1997). Movement, activity, and action: The role of knowledge in the perception of motion. *Phil. Trans. Royal Society London B*.

[12] Bobick, A. F. and Davis, J. W. (2001). The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267.

[13] Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *International Conference on Pattern Recognition*, pages 77–77. IEEE Computer Society Press.

[14] Bregonzio, M., Gong, S., and Xiang, T. (2009). Recognising action as clouds of space-time interest points. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1948–1955. IEEE.

[15] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):1–27.

[16] Chen, H.-S., Chen, H.-T., Chen, Y.-W., and Lee, S.-Y. (2006). Human action recognition using star skeleton. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 171–178. ACM.

[17] Cherla, S., Kulkarni, K., Kale, A., and Ramasubramanian, V. (2008). Towards fast, view-invariant human action recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE.

[18] Chin, T.-J., Wang, L., Schindler, K., and Suter, D. (2007). Extrapolating learned manifolds for human activity recognition. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 1, pages I–381. IEEE.

[19] Chomat, O., Martin, J., and Crowley, J. L. (2000). A probabilistic sensor for the perception and the recognition of activities. In *Computer Vision-ECCV 2000*, pages 487–503. Springer.

[20] Chris, M. and Schuetze, H. (1999). Foundations of statistical natural language processing.

[21] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[22] Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. 2:265–292.

[23] Danafar, S. and Gheissari, N. (2007). Action recognition for surveillance applications using optic flow and svm. In *Computer Vision–ACCV 2007*, pages 457–466. Springer.

[24] De la Torre, F., Hodgins, J. K., Montano, J., and Valcarcel, S. (2009). Detailed human data acquisition of kitchen activities: the cmu-multimodal activity database (cmu-mmac). In *Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research, in conjuction with CHI 2009.*

[25] Doherty, M. K. (2007). *Gene prediction with conditional random fields*. PhD thesis, Broad Institute.

[26] Dollár, P., Rabaud, V., Cottrell, G., and Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72. IEEE.

[27] Efros, A., Berg, A. C., Mori, G., Malik, J., et al. (2003). Recognizing action at a distance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726–733. IEEE.

[28] Erdogan, H. (2010). Sequence labeling: Generative and discriminative approaches hidden markov models, conditional random field and structured svm. In *Tutorial at International Conference on Machine Learning and Applications*, Washington D.C., USA. IEEE.

[29] Escobar, M.-J., Masson, G. S., Vieville, T., and Kornprobst, P. (2009). Action recognition using a bio-inspired feedforward spiking network. *International Journal of Computer Vision*, 82(3):284–301.

[30] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.

[31] Fanti, C., Zelnik-Manor, L., and Perona, P. (2005). Hybrid models for human motion recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 1166–1173. IEEE.

[32] Farhadi, A. and Tabrizi, M. K. (2008). Learning to recognize activities from the wrong view point. In *Computer Vision–ECCV 2008*, pages 154–166. Springer.

[33] Farhadi, A., Tabrizi, M. K., Endres, I., and Forsyth, D. (2009). A latent model of discriminative aspect. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 948–955. IEEE.

[34] Fathi, A. and Mori, G. (2008). Action recognition by learning mid-level motion features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[35] Filipovych, R. and Ribeiro, E. (2008). Learning human motion models from unsegmented videos. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE.

[36] Forsyth, D. A., Arikan, O., and Ikemoto, L. (2006). *Computational Studies of Human Motion: Tracking and Motion Synthesis*. Now Publishers Inc.

[37] Friedman, J. (1996). Another approach to polychotomous classifcation. *Dept. Statist., Stanford Univ., Stanford, CA, USA, Tech. Rep.*

[38] Fürnkranz, J., Hüllermeier, E., Mencía, E. L., and Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153.

[39] Gaidon, A., Marszalek, M., and Schmid, C. (2009). Mining visual actions from movies. In Cavallaro, A., Prince, S., and Alexander, D., editors, *British Machine Vision Conference*, pages 125.1–125.11, Londres, United Kingdom. British Machine Vision Association, BMVA Press. Page web de l'article : http://lear.inrialpes.fr/pubs/2009/GMS09/.

[40] Gavrila, D. M. (1999). The visual analysis of human movement: A survey. In *Computer Vision and Image Understanding*, volume 73, pages 82–98. Academic Press.

[41] Gilbert, A., Illingworth, J., and Bowden, R. (2008). Scale invariant action recognition using compound features mined from dense spatio-temporal corners. In *Computer Vision–ECCV 2008*, pages 222–233. Springer.

[42] Grundmann, M., Meier, F., and Essa, I. (2008). 3d shape context and distance transform for action recognition. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

[43] Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, C. (2008). Automatic Face Naming with Caption-based Supervision. In *CVPR 2008 - IEEE Conference on Computer Vision & Pattern Recognition*, pages 1–8, Anchorage, United States. IEEE Computer society.

[44] Har-Peled, S., Roth, D., and Zimak, D. (2002). Constraint classification: A new approach to multiclass classification. In *Algorithmic Learning Theory*, pages 365–379. Springer.

[45] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer.

[46] Hoai, M., Lan, Z.-Z., and De la Torre, F. (2011). Joint segmentation and classification of human actions in video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

[47] Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916.

[48] Ikizler, N., Cinbis, R. G., and Duygulu, P. (2008a). Human action recognition with line and flow histograms. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

[49] Ikizler, N., Cinbis, R. G., Pehlivan, S., and Duygulu, P. (2008b). Recognizing actions from still images. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

[50] Ikizler, N. and Duygulu, P. (2009). Histogram of oriented rectangles: A new pose descriptor for human action recognition. *Image and Vision Computing*, 27(10):1515–1526.

[51] Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR 2010 - 23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311.

[52] Jhuang, H., Serre, T., Wolf, L., and Poggio, T. (2007). A biologically inspired system for action recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. Ieee.

[53] Jia, K. and Yeung, D.-Y. (2008). Human action recognition using local spatio-temporal discriminant embedding. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[54] Jiang, H. and Martin, D. R. (2008). Finding actions using shape flows. In *Computer Vision–ECCV 2008*, pages 278–292. Springer.

[55] J.K.Aggarwal, Q. (1999). Tracking human motion in a structured environment using a distributed camera system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:1241–1247.

[56] Joachims, T. (2005). A support vector method for multivariate performance measures. In *ICML*, Bonn, Germany. IEEE.

[57] Joachims, T. (2008). SVM$^{\text{struct}}$: Support vector machine for complex output 3.10.

[58] Jurafsky, D. and Martin, J. H. (2008). *Speech and language processing*. Prentice Hall, New Jersey, USA, 2 edition.

[59] K. Evang, V. Basile, G. C. and Bos, J. (2013). Elphant: Sequence labeling for word and sentence segmentation. In *Proceeding of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1426. IEEE.

[60] Kadir, T. and Brady, M. (2003). Scale saliency: A novel approach to salient feature and scale selection.

[61] Ke, Y., Sukthankar, R., and Hebert, M. (2005). Efficient visual event detection using volumetric features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 166–173. IEEE.

[62] Ke, Y., Sukthankar, R., and Hebert, M. (2007a). Event detection in crowded videos. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[63] Ke, Y., Sukthankar, R., and Hebert, M. (2007b). Spatio-temporal shape and flow correlation for action recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[64] Kellokumpu, V., Zhao, G., and Pietikäinen, M. (2008). Human activity recognition using a dynamic texture based method. In *BMVC*, volume 1, page 2.

[65] Kim, T.-K. and Cipolla, R. (2009). Canonical correlation analysis of video volume tensors for action categorization and detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(8):1415–1428.

[66] Kishida, K. (2005). *Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments*. National Institute of Informatics Tokyo, Japan.

[67] Klaser, A., Marszałek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association.

[68] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *NAACL-HLT*.

[69] Laptev, I. (2005). On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123.

[70] Laptev, I., Caputo, B., Schüldt, C., and Lindeberg, T. (2007). Local velocity-adapted motion events for spatio-temporal recognition. *Computer Vision and Image Understanding*, 108(3):207–229.

[71] Laptev, I., Marszałek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[72] Laptev, I. and Pérez, P. (2007). Retrieving actions in movies. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[73] Lin, Z., Jiang, Z., and Davis, L. S. (2009). Recognizing actions by shape-motion prototype trees. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 444–451. IEEE.

[74] Liu, J., Ali, S., and Shah, M. (2008). Recognizing human actions using multiple features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[75] Liu, J., Luo, J., and Shah, M. (2009). Recognizing realistic actions from videos in the wild. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1996–2003. IEEE.

[76] Liu, J. and Shah, M. (2008). Learning human actions via information maximization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[77] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[78] Loza Mencía, E. and Furnkranz, J. (2008). Pairwise learning of multilabel classifications with perceptrons. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2899–2906. IEEE.

[79] Lu, W.-L. and Little, J. J. (2006). Simultaneous tracking and action recognition using the pca-hog descriptor. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pages 6–6. IEEE.

[80] L.Wang, W. and T.Tan (2002). A new attempt to gait-based human identification. In *IEEE on International Conference on Pattern Recognition*, pages 115–118.

[81] Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

[82] Mao, Q. and Tsang, I. W.-H. (2013). A feature selection method for multivariate performance measures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(9):2051–2063.

[83] Masoud, O. and Papanikolopoulos, N. (2003). A method for human action recognition. *Image and Vision Computing*, 21(8):729–743.

[84] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., and Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

[85] Messing, R., Pal, C., and Kautz, H. (2009). Activity recognition using the velocity histories of tracked keypoints. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 104–111. IEEE.

[86] Mikolajczyk, K. and Uemura, H. (2008). Action recognition with motion-appearance vocabulary forest. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[87] Morgan, R., Sutcliffe, C., and O'neill, W. (2004). Density analysis of direct metal laser re-melted 316l stainless steel cubic primitives. *Journal of Materials Science*, 39(4):1195–1205.

[88] Nguyen, N. and Guo, Y. (2007). Comparison of sequence labeling algorithms and extensions. In *ICML*, Corvallis, USA.

[89] Niebles, J. C., Wang, H., and Fei-Fei, L. (2008). Unsupervised learning of human action categories using spatial-temporal words. *International journal of computer vision*, 79(3):299–318.

[90] Nilsson, P., Larhed, M., and Hallberg, A. (2001). Highly regioselective, sequential, and multiple palladium-catalyzed arylations of vinyl ethers carrying a coordinating auxiliary: an example of a heck triarylation process. *Journal of the American Chemical Society*, 123(34):8217–8225.

[91] Ning, H., Hu, Y., and Huang, T. S. (2007). Searching human behaviors using spatial-temporalwords. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 6, pages VI–337. IEEE.

[92] Nowozin, S., Bakir, G., and Tsuda, K. (2007). Discriminative subsequence mining for action classification. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[93] Nowozin, S. and Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Found. Trends. Comput. Graph. Vis.*, 6(3-4):185–365.

[94] Ogale, A. S., Karapurkar, A., and Aloimonos, Y. (2007). View-invariant modeling and recognition of human actions using grammars. In *Dynamical vision*, pages 115–126. Springer.

[95] Ogata, T., Christmas, W., Kittler, J., and Ishikawa, S. (2006). Improving human activity detection by combining multi-dimensional motion descriptors with boosting. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 295–298. IEEE.

[96] Oikonomopoulos, A., Pantic, M., and Patras, I. (2009). Sparse b-spline polynomial descriptors for human activity recognition. *Image and Vision Computing*, 27(12):1814–1825.

[97] Oikonomopoulos, A., Patras, I., and Pantic, M. (2005). Spatiotemporal salient points for visual recognition of human actions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(3):710–719.

[98] Oshin, O., Gilbert, A., Illingworth, J., and Bowden, R. (2008). Spatio-temporal feature recognition using randomised ferns. In *The 1st International Workshop on Machine Learning for Vision-based Motion Analysis (MVLMA'08)*.

[99] Patron-Perez, A., Reid, I., Patron, A., and Reid, I. (2007). A probabilistic framework for recognizing similar actions using spatio-temporal features. In *BMVC*, pages 1–10.

[100] Petterson, J. and Caetano, T. (2010). Reverse multi-label learning. In *NIPS*, Whistler, Canada. IEEE.

[101] Platt, J. C., Cristianini, N., and Shawe-Taylor, J. (1999). Large margin dags for multiclass classification. In *nips*, volume 12, pages 547–553.

[102] Poppe, R. (2010). A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990.

[103] Poppe, R. and Poel, M. (2008a). Body-part templates for recovery of 2d human poses under occlusion. In *Articulated Motion and Deformable Objects*, pages 289–298. Springer.

[104] Poppe, R. and Poel, M. (2008b). Discriminative human action recognition using pairwise csp classifiers. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–6. IEEE.

[105] P.Turaga, A. V. and R.Chellappa (2008). Statistical analysis on stiefel and grassmann manifolds with applications in computer vision. In *in IEEE conference on Computer Vision and Pattern Recognition*.

[106] P.Turaga, R.Chellappa, V. and O.Udrea (2008). Machine recognition of human activities: A survey. In *in IEEE Transactions on Circuits and Systems for Video Technology*.

[107] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE Proc.*, 77:257–286.

[108] Ragheb, H., Velastin, S., Remagnino, P., and Ellis, T. (2008). Human action recognition using robust power spectrum features. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 753–756. IEEE.

[109] Ramanan, D. (2006). Learning to parse images of articulated bodies. In *Advances in neural information processing systems*, pages 1129–1136.

[110] Ranjbar, M., Lan, T., Wang, Y., Robinovitch, S. N., Li, Z.-N., and Mori, G. (2013). Optimizing nondecomposable loss functions in structured prediction. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):911–924.

[111] Ranjbar, M., Vahdat, A., and Mori, G. (2012). Complex loss optimization via d-ual decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2304–2311. IEEE.

[112] Rapantzikos, K., Avrithis, Y., and Kollias, S. (2007). Spatiotemporal saliency for event detection and representation in the 3d wavelet domain: potential in human action recognition. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 294–301. ACM.

[113] Rapantzikos, K., Avrithis, Y., and Kollias, S. (2009). Dense saliency-based spatiotemporal feature points for action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1454–1461. IEEE.

[114] Read, J. (2008). A pruned problem transformation method for multi-label classification. In *In: Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS*, pages 143–150.

[115] Rodriguez, M. D., Ahmed, J., and Shah, M. (2008). Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[116] Roller, B. T. C. G. D. (2004). Max-margin markov networks. *Advances in neural information processing systems*, 16:25.

[117] Rosales, R. (1998). Recognition of human action using moment-based.

[118] Rosenfeld, N., Meshi, O., Tarlow, D., and Globerson, A. (2014). Learning structured models with the auc loss and its generalizations. In *AISTATS*, pages 841–849. Citeseer.

[119] Savarese, S., DelPozo, A., Niebles, J. C., and Fei-Fei, L. (2008). Spatial-temporal correlatons for unsupervised action classification. In *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*, pages 1–8. IEEE.

[120] Schindler, K. and Van Gool, L. (2008). Action snippets: How many frames does human action recognition require? In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[121] Schüldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE.

[122] Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–360. ACM.

[123] Smith, N. (2009). Structured prediction for natural language processing. In *ICML tutorial*.

[124] Smith, P., Lobo, N. D. V., and Shah, M. (2005). Temporalboost for event recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 733–740. IEEE.

[125] Song, Y., Goncalves, L., and Perona, P. (2003). Unsupervised learning of human motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):814–827.

[126] Sun, J., Wu, X., Yan, S., Cheong, L.-F., Chua, T.-S., and Li, J. (2009). Hierarchical spatio-temporal context modeling for action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2004–2011. IEEE.

[127] Taskar, B., Guestrin, C., and Koller, D. (2003a). Max-margin Markov networks. In *Proc. NIPS*.

[128] Taskar, B., Guestrin, C., and Koller, D. (2003b). Max-margin markov networks. In *NIPS*.

[129] T.B.Moeslund, A. and V.Kruger (2006). A survey of advances in vision-based human motion capture and analysis. In *Computer Vision and Image Understanding*, volume 104, pages 90–127. Academic Press.

[130] Tenorth, M., Bandouch, J., and Beetz, M. (2009). The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV2009*.

[131] Thurau, C. and Hlaváč, V. (2008). Pose primitive based human action recognition in videos or still images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[132] Tran, D., Sorokin, A., and Forsyth, D. (2008). Human activity recognition with metric learning. *Computer Vision–ECCV 2008*, pages 548–561.

[133] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM.

[134] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.

[135] Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *Machine learning: ECML 2007*, pages 406–417. Springer Berlin Heidelberg.

[136] Turaga, P., Veeraraghavan, A., and Chellappa, R. (2008). Statistical analysis on stiefel and grassmann manifolds with applications in computer vision. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[137] Uemura, H., Ishikawa, S., and Mikolajczyk, K. (2008). Feature tracking and motion compensation for action recognition. In *BMVC*, pages 1–10.

[138] Vedaldi, A. (2011). A MATLAB wrapper of SVM$^{\text{struct}}$.

[139] Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms.

[140] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE.

[141] Vitaladevuni, S. N., Kellokumpu, V., and Davis, L. S. (2008). Action recognition using ballistic dynamics. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[142] V.Kruger, D. A. and C.Geib (2007). The meaning of action : a review on action recognition and mapping. In *Advanced Robotics*, volume 21, pages 1473–1501.

[143] Wang, H., Ullah, M. M., Klaser, A., Laptev, I., and Schmid, C. (2009). Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, pages 124–1. BMVA Press.

[144] Wang, L. and Suter, D. (2006). Informative shape representations for human action recognition. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 1266–1269. IEEE.

[145] Wang, L. and Suter, D. (2007). Learning and matching of dynamic shape manifolds for human action recognition. *Image Processing, IEEE Transactions on*, 16(6):1646–1661.

[146] Wang, L. and Suter, D. (2008). Visual learning and recognition of sequential data manifolds with applications to human movement analysis. *Computer Vision and Image Understanding*, 110(2):153–172.

[147] Wang, L. and Yang, R. (2011). Global stereo matching leveraged by sparse ground control points. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3033–3040. IEEE.

[148] Wang, S. B., Quattoni, A., Morency, L.-P., and Demirdjian, D. (2006a). Hidden conditional random fields for gesture recognition. In *Proc. CVPR*, pages 2:1521–1527.

[149] Wang, Y., Jiang, H., Drew, M. S., Li, Z.-N., and Mori, G. (2006b). Unsupervised discovery of action classes. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1654–1661. IEEE.

[150] Wang, Y. and Mori, G. (2009a). Human action recognition by semilatent topic models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1762–1774.

[151] Wang, Y. and Mori, G. (2009b). Learning a discriminative hidden part model for human action recognition. In *Advances in Neural Information Processing Systems*, pages 1721–1728.

[152] Wang, Y. and Mori, G. (2009c). Max-margin hidden conditional random fields for human action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 872–879. IEEE.

[153] Weinland, D. and Boyer, E. (2008). Action recognition using exemplar-based embedding. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE.

[154] Weinland, D., Boyer, E., and Ronfard, R. (2007). Action recognition from arbitrary views using 3d exemplars. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE.

[155] Weinland, D., Ronfard, R., and Boyer, E. (2005). Motion history volumes for free viewpoint action recognition. In *Workshop on modeling People and Human Interaction (PHI'05)*.

[156] Weinland, D., Ronfard, R., and Boyer, E. (2006). Automatic discovery of action taxonomies from multiple views. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1639–1645. IEEE.

[157] Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical report, Citeseer.

[158] Willems, G., Tuytelaars, T., and Van Gool, L. (2008). An efficient dense and scale-invariant spatio-temporal interest point detector. In *Computer Vision–ECCV 2008*, pages 650–663. Springer.

[159] Wong, S.-F. and Cipolla, R. (2007). Extracting spatiotemporal interest points using global information. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[160] Wong, S.-F., Kim, T.-K., and Cipolla, R. (2007). Learning motion categories using both semantic and structural information. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE.

[161] Xu, G. et al. (2007). Viewpoint insensitive action recognition using envelop shape. In *Computer Vision–ACCV 2007*, pages 477–486. Springer.

[162] Yan, P., Khan, S. M., and Shah, M. (2008). Learning 4d action feature models for arbitrary view action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE.

[163] Yao, B. and Zhu, S.-C. (2009). Learning deformable action templates from cluttered videos. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1507–1514. IEEE.

[164] Yilmaz, A. and Shah, M. (2008). A differential geometric approach to representing the human actions. *Computer Vision and Image Understanding*, 109(3):335–351.

[165] Yilmaz, E. and Aslam, J. A. (2006). Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111. ACM.

[166] Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007). A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM.

[167] Zare-Borzeshi, E., Perez-Concha, O., Xu, R. Y. D., and Piccardi, M. (2013). Joint action segmentation and classification by an extended hidden Markov model. *IEEE Signal Processing Letters*, 20(12):1207–1210.

[168] Zhang, K. and Fan, W. (2008). Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems*, 14(3):299–326.

[169] Zhang, M.-L. and Zhou, Z.-H. (2007). Multi-label learning by instance differentiation. In *AAAI*, volume 7, pages 669–674.

[170] Zhang, X. and Zou, J. (2010). A structural svm approach for reference parsing. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, Washington D.C., USA. IEEE.

[171] Zhang, Z., Hu, Y., Chan, S., and Chia, L.-T. (2008). Motion context: A new representation for human action recognition. *Computer Vision–ECCV 2008*, pages 817–829.

[172] Zhao, L., Wang, X., Sukthankar, G., and Sukthankar, R. (2010). Motif discovery and feature selection for crf-based activity recognition. In *ICPR*, pages 3826–3829.

[173] Zhao, Z. and Elgammal, A. (2008). Human activity recognition from frames spatiotemporal representation. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

Appendix A

Algorithm Correctness

Proposition. Given ground-truth label at frame t $y_t^g = 0$, the best predicted sequence $y_{1:t}$ with FP false positives, FN false negatives, and ending in $y_t = 0$, noted as $\psi(FP, FN, y_t = 0)$, is given by $\text{argmax}(s([\psi(FP, FN, y_{t-1} = 0), 0]), s([\psi(FP, FN, y_{t-1} = 1), 0]))$.

Proof. The proof is a simple argument by contradiction. Let us note $\psi(FP, FN, y_t = 0)$ as $[y^*_{1:t-1}, 0]$. Since y_t^g and y_t concur, sequence $y^*_{1:t-1}$ must contain FP false positives and FN false negatives.

Now, if $\psi(FP, FN, y_t = 0) \neq \text{argmax}(s([\psi(FP, FN, y_{t-1} = 0), 0]), s([\psi(FP, FN, y_{t-1} = 1), 0]))$, then $y^*_{1:t-1}$ must differ from both $\psi(FP, FN, y_{t-1} = 0)$ and $\psi(FP, FN, y_{t-1} = 1)$. This conflicts with the definition of $\psi(FP, FN, y_{t-1})$.

The proof for the other three combinations of y_t^g and y_t is analogous $\square$.

Index