Minimum-risk Sequence Alignment
for the Alignment and Recognition of
Action Videos

by

Zhen Wang

A thesis submitted in partial fulfilment for the
degree of Doctor of Philosophy

at the
School of Electrical and Data Engineering
Faculty of Engineering and Information Technology
University of Technology Sydney

July 2018
Declaration of Authorship

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This research was supported by an Australian Government Research Training Program (RTP).

Signed: ____________________________  Production Note:  
Signature removed prior to publication.

Date: ______________________________  18/07/2018
Abstract

Temporal alignment of videos is an important requirement of tasks such as video comparison, analysis and classification. In the context of action analysis and action recognition, the main guiding element for the temporal alignment are the human actions depicted in the videos. While well-established alignment algorithms such as dynamic time warping are available, they still heavily rely on basic linear cost models and heuristic parameter tuning. Inspired by the success of the hidden Markov support vector machine for pairwise alignment of protein sequences, in this thesis we present a novel framework which combines the flexibility of a pair hidden Markov model (PHMM) with the effective parameter training of the structural support vector machine (SSVM). The framework extends the scoring function of SSVM to capture the similarity of two input frame sequences and introduces suitable feature and loss functions. During learning, we leverage these loss functions for regularised empirical risk minimisation and effective parameter selection.

We have carried out extensive experiments with the proposed technique (nicknamed as EHMM-SSVM) against state-of-the-art algorithms such as dynamic time warping (DTW) and generalized canonical time warping (GCTW) on pairs of human actions from four well-known datasets. The results show that the proposed model has been able to outperform the compared algorithms by a large margin in terms of alignment accuracy.

In the second part of this thesis we employ our alignment approach to tackle the task of human action recognition in video. This task is highly challenging due to the substantial variations in motion performance, recording settings and inter-personal differences. Most current research focuses on the extraction of effective features and the design of suitable classifiers. Conversely, in this thesis we tackle this problem by a dissimilarity-based approach where classification is performed in terms of minimum distance from templates and where the distance is based on the score of our alignment model, the EHMM-SSVM. In turn, the templates are chosen by means of prototype selection techniques from the available samples of each class. Experimental results over two popular human action datasets
have showed that the proposed approach has been capable of achieving an accuracy higher than many existing methods and comparable to a state-of-the-art action classification algorithm.
Acknowledgements

I would like to take this opportunity to acknowledge the enormous support from my supervisor, Prof. Massimo Piccardi, the very useful suggestions from my group and the friendly environment of our lab especially during coffees, lunches and dinners.
Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iv

List of Figures viii

List of Tables x

1 Introduction 1
   1.1 Background on temporal alignment 3
   1.2 Minimum-risk temporal alignment 5
   1.3 Classification based on alignment costs 5
   1.4 Contributions 7
   1.5 Publications 7
   1.6 Thesis Outline 8

2 Review of Related Works 9
   2.1 Dynamic Time Warping 10
   2.2 Generalized Canonical Time Warping 13
      2.2.1 Canonical Correlation Analysis (CCA) 14
      2.2.2 Combination of CCA and DTW 16
   2.3 Hidden Markov model 18
   2.4 Pair hidden Markov model 20
   2.5 Prototype Selection 23
   2.6 Support vector machines 26
      2.6.1 Binary SVM 27
      2.6.2 Soft-margin SVM 30
   2.7 Multi-Class SVM 31
      2.7.1 One-vs-all classifier 32
      2.7.2 One-vs-one classifier 33
      2.7.3 Directed Acyclic Graph-support vector machines 34
2.8 The Structural Support Vector Machine ................................. 35

3 Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos 40
3.1 Introduction .................................................................. 40
3.2 The Proposed Model: PHMM-SSVM ................................. 42
  3.2.1 Pair Hidden Markov Model ................................ 42
  3.2.2 Structural SVM ............................................... 44
  3.2.3 Integration ...................................................... 45
  3.2.4 Parameter Vector and Feature Function ......................... 46
  3.2.5 Loss Functions .................................................. 47
  3.2.6 Most-Violated Constraints ...................................... 47
3.3 Experiments .................................................................... 50
  3.3.1 Results on the Weizmann Dataset ............................... 51
  3.3.2 Results on the Olympic Sports Dataset ........................ 52
3.4 Conclusion ...................................................................... 53

4 The Extended Hidden Markov Support Vector Machine and its Application to the Alignment of Human Actions 54
4.1 Introduction .................................................................. 54
4.2 Extended Hidden Markov Model ...................................... 55
4.3 The Proposed Model ...................................................... 57
  4.3.1 Scoring Function .................................................. 58
  4.3.2 Dissimilarity Functions .......................................... 59
  4.3.3 Loss Function and Loss-Augmented Inference .................. 59
  4.3.4 Interpolation of the Ground-Truth Alignments Based on Key-Frames 61
4.4 Experimental Results ..................................................... 64
  4.4.1 UCF101 Dataset ................................................. 64
  4.4.2 MSR Daily Activity 3D Dataset ................................ 65
4.5 Conclusion ...................................................................... 67

5 Dissimilarity-Based Action Recognition with an Extended Hidden Markov Support Vector Machine 69
5.1 Introduction .................................................................. 69
5.2 EHMM-SSVM distance .................................................... 71
5.3 Prototype selection with EHMM-SSVM ............................... 73
5.4 Classification .................................................................. 74
5.5 Experiments and Discussion ............................................. 77
  5.5.1 Results with the k-NN classifier on the KTH dataset ........ 77
  5.5.2 Results with the k-NN classifier on the Olympic Sports dataset ... 79
  5.5.3 Results with the DAGSVM classifier on both datasets ....... 79
5.6 Conclusion ..................................................................... 81

6 Conclusion 83
Contents

Appendices 84

Bibliography 85
List of Figures

1.1 Example of the Euclidean distances between two sequences. ........................................... 2
1.2 DTW distances between two sequences. ............................................................................. 3
1.3 Example of optimal warping path produced by DTW. ...................................................... 4
1.4 Example of extended hidden Markov model. Latent variable $y_i$ stores two paired indexes from sequences $s$ and $t$. .......................................................... 4

2.1 Illustration of paths of index pairs for some sequence $s$ of length $N = 9$ and some sequence $t$ of length $M = 7$. (A) Admissible warping path satisfying the conditions (i), (ii), and (iii) of Definition 2.1. (B) Boundary condition (i) is violated. (C) Monotonicity condition (ii) is violated. (D) Step size condition (iii) is violated. .......................................................... 11
2.2 Example of cost matrices in the DTW algorithm. .............................................................. 12
2.3 Hidden Markov model (HMM) graphical model. .............................................................. 18
2.4 PHMM state diagram. ........................................................................................................ 22
2.5 A hyperplane separating two classes C1 and C2. ............................................................. 27
2.6 Geometric margins at each side of the hyperplane. ......................................................... 28
2.7 A single "outlier" can affect the separating hyperplane significantly. ............................. 30
2.8 Example of data from three hypothetical classes. ............................................................ 32
2.9 Examples of one-vs-all classifier. .................................................................................... 32
2.10 Examples of one-vs-one classifier. .................................................................................. 33
2.11 An example of DAGSVM ............................................................................................... 34
2.12 Illustration of a natural language parsing model. The graphics are reproduced from [37]. .......................................................... 37

3.1 PHMM state diagram. Transitions "Insert a gap" between $I_s$ and $I_t$ are prohibited. .......................................................... 46
3.2 Sample actions ................................................................................................................ 51
3.3 Example of ground-truth and predicted alignments from the Olympic Sports dataset: top two rows): six matched key-frames from two clean-and-jerk sequences. The first row is used as template and the frames of the second row show the ground-truth alignment; third row): alignment predicted by the proposed PHMM-SSVM; bottom two rows): alignments predicted by GCTW and DTW. The superimposed rectangles visually highlight the alignment errors. .......................................................... 52
4.1 The extended hidden Markov model for sequence alignment (represented as an undirected graphical model). .......................................................... 56

4.2 Main steps of the proposed approach: a) training (with regularized empirical risk minimization, i.e., structural SVM - Section 3.2.6); inference (Section 4.2). .......................................................... 56

4.3 Example of the interpolations on key-frames ground-truth alignments on the pairs of (1, 1), (5, 6), (9, 11), (12, 15) from UCF101 dataset. ......................... 62

4.4 Example of alignment obtained with the proposed approach for two videos of action “body weight squats” from UCF101. ................................. 66

5.1 Left: Example of frame-by-frame distances. Right: Example of dynamic time warping. .......................................................... 70

5.2 Examples of EHMM-SSVM alignment from the Olympic Sport dataset. Top two rows: 14 matches and inserted gaps for two paired “clean-and-jerk” sequences. Bottom two rows: 16 matches and inserted gaps for two paired “tennis-serve” sequences (the gaps are inserted only on the second sequence because of its slower execution). .......................................................... 71

5.3 The EHMM-SSVM as a graphical model. .................................................. 72

5.4 Sample actions in the Olympic dataset (top four rows) and KTH dataset (bottom row). .......................................................... 76

5.5 The EHMM-SSVM distance matrix between action instances in KTH dataset. Darker colours denote higher similarity. ........................................ 78
List of Tables

2.1 Numerical example of the cost matrix and the accumulated cost matrix determined during the formation of the optimal warping path. (A) Each cell represents the distance between elements $s_i$ and $t_j$ and is computed as cost function $C(s_i, t_j) = |s_i - t_j|$. (B) The accumulated cost matrix is obtained using Definition 2.2. ........................................... 13

2.2 Transition probabilities table. ........................................... 22

3.1 Transition probabilities table. ........................................... 46

3.2 Alignment accuracy for action “jump” in the Weizmann dataset. .......... 51

3.3 Alignment accuracy for action “clean-and-jerk” in the Olympic Sports dataset. 53

4.1 Transition probabilities table. ........................................... 56

4.2 Experimental results for action “body weight squats” from the UCF101 dataset. ........................................... 65

4.3 Experimental results for action “stand up” from the MSR Daily Activity 3D dataset. ........................................... 65

4.4 Example data from the MSR Daily Activity 3D dataset. .................... 67

4.6 Example of skeleton alignment from action “stand up” in the MSR Daily Activity 3D dataset. The arrows highlight noticeable inaccuracies. .......... 68

5.1 Example of EHMM-SSVM distances between a walking sample in the KTH dataset and prototypes from the various classes. Darker colours denote higher similarity. ........................................... 76

5.2 Accuracy with the $k$-NN classifier on the KTH dataset. .................... 79

5.3 Accuracy in the second experiment on the Olympic Sports dataset. ........ 80

5.4 Accuracy with the DAGSVM classifier on the KTH dataset. ............... 80

5.5 Accuracy with the DAGSVM classifier on the Olympic Sports dataset. ... 81
Chapter 1

Introduction

Sequential data play a paramount role in fields as diverse as finance, computer vision, speech recognition, natural language processing, weather forecasting and bioinformatics, for a variety of tasks that include automated classification, clustering, anomaly detection and more. Examples of sequential data range from time series of share prices to videos, speech utterances, text and DNA sequences. Their main difference with respect to “ordinary” data is that they are inherently ordered into sequences, and that the time stamps or indexes in the sequence need to be taken into account during the analysis (or, if dismissed, would lead to poor modelling).

One of the immediate problems that arise when dealing with sequential data is how to “meaningfully” compare two or more sequences. In some cases, the events described in two sequences may appear similar in qualitative terms, but not start at the same time, have different overall duration, and progress at different speed. In these cases, orderly frame-to-frame comparisons would most likely fail to recognise the similarity. What is required to mollify this problem is to be able to align the sequences by their actual “phases/stages”, and compare them from the framework of this normalization. Fig. 1.1 shows an example of two sequences where the main phases can be easily identified and placed into correspondence. In the literature, this problem is formally known as sequence “alignment” or “warping” and is core to tasks such as sequence analysis, clustering and classification. The problem is often framed as the alignment of two given sequences, where the first is used
Chapter 1. *Introduction*

![Figure 1.1: Example of the Euclidean distances between two sequences.](image)

as the reference and the second is aligned, or “warped”, onto the first. Intuitively, the dissimilarity between the two sequences can be described as the amount of warping effort or atomic warping operations required to warp the second sequences onto the first. Sequence warping has been employed extensively in the context of gene finding [2][3], text mining[4], finance[5], and video, where the goal of sequence alignment translates into finding corresponding frames of two given videos. In this thesis, we propose an original solution to this problem by employing a contemporary machine learning approach known as structural risk minimisation.

Being able to effectively align two videos can prove key to recognising the human activities depicted in the videos. Automated recognition of human activities is a ubiquitous problem in computer vision for tasks such as human-computer interaction, user interface design, robot navigation, video surveillance etc. Traditionally, research on activity recognition has focussed on the extraction of suitable features from the videos and the design of accurate classifiers. Conversely, in this thesis we explore a dissimilarity-based approach, where classification is performed in terms of minimum distance between the target video and template videos from the various classes. Consistently with our overall thesis, the distance between the target and a template is suitably defined in terms of alignment effort. To minimise the number of comparisons required and improve the classification accuracy, we reduce the number of templates per class by well-proven “prototype selection” techniques.
1.1 Background on temporal alignment

Dynamic time warping (DTW) is likely the most well-known temporal alignment algorithm [1], which has found application for tasks as diverse as gesture recognition [6], speech processing [7], classification of genomic signals [8] and many others. DTW finds an optimal, non-linear alignment path between two time series by examining local alignment costs [1]. An illustrative example is given in figure 1.2. DTW returns a set of index correspondences in the two sequences, that is the optimal temporal mapping (also called the optimal warping path). A more detailed example is shown in figure 1.3 where the two 1-D sequences are plotted as curves.

To main limitation of the standard DTW is its quadratic computational cost, proportional to the product of the lengths of the two sequences. To mollify this issue, various variants have been proposed. Windowing DTW restricts the corresponding frames to fall within a given window [9]. Slope-weight DTW restricts the search to paths within a given slope [10]. Keogh and Pazzani [11] have used derivatives of the original signals to improve the alignment before applying DTW. Amongst the many variants, the state-of-the-art algorithm is likely generalized canonical time warping (GCTW) that iteratively applies canonical correlation analysis (CCA) alongside DTW [12]. GCTW has reported the highest accuracy over a variety of video benchmarks [12, 13]. While the DTW variants have improved DTW and specialised it for different problems, they are still imprisoned in a fixed, arbitrary cost model for computing the distance of the two sequences.
Figure 1.3: Example of optimal warping path produced by DTW.

Figure 1.4: Example of extended hidden Markov model. Latent variable $y_i$ stores two paired indexes from sequences $s$ and $t$. 
Chapter 1. Introduction

1.2 Minimum-risk temporal alignment

To amend this issue in the DTW family, we will discuss a novel alignment algorithm that 
adopts an extended hidden Markov model and learns the model’s parameters from a set 
of manually-aligned sequence pairs. This extended HMM is a graphical model consisting 
of the two sequences of measurements and a Markov chain of latent states that encodes 
their alignment path (see figure 1.4). It is a so-called “generative” sequential model that 
provides a full probabilistic treatment of the alignment problem [14]. Rather than learning 
its parameters under a conventional maximum-likelihood framework, we will make use of 
the more performing and current structural support vector machine [15]. This new model 
- named extended hidden Markov support vector machine (EHMM-SSVM) hereafter - 
supports flexible cost models (potentially non-linear, using kernels) and tends to achieves 
higher alignment accuracy thanks to its minimum-risk parameter estimation.

1.3 Classification based on alignment costs

The alignment of two sequences provides, as by-product, a “cost”, which can be interpreted 
as an indicator of the dissimilarity of the two sequences. Based on such costs, one 
could build a dissimilarity-based sequence classifier. Classic dissimilarity-based classifiers 
construct decision rules based on the magnitude of the difference between two objects. 
Given a set of class-labelled templates, classification can thereby be provided by the use of 
a minimum-distance classifier (e.g., k-nearest neighbours), with the decision being based 
on the distances between the test sequence and the templates from the various classes. 
Dissimilarity-based classification has reported remarkable accuracy [16–20], and has proved 
very successful for the classification of non-vector objects such as graphs and strings [16–18]. Therefore, we expect it to be promising also for the classification of complex sequential 
data such as human actions.

The more popular siblings of distance-based classifiers are known as feature-based classifiers. Such classifiers perform classification based on a “feature vector” which can be obtained in various ways from the original objects. Interestingly, the distances between 
the object and the class templates can also be treated as a a particular feature vector, or
embedding, and used in conjunction with any feature-based classifiers. For instance, given a set of templates \( x_1, \ldots, x_n \), to classify a new vector \( x \) with a support vector machine (SVM), one would first extract the dissimilarity-based feature vector \( D(x) = [D(x_1, x), \ldots, D(x_n, x)] \) and then use it with the scoring function of the SVM, \( f(x) = w^T D(x) + b \).

Since both the classification approaches described above require a measure of dissimilarity, we will use our extended hidden Markov support vector machine (EHMM-SSVM) as a more sophisticated distance measure than crude frame-to-frame distances. The EHMM-SSVM provides major advantages over conventional alignment algorithms: 1) flexible loss functions that facilitate tuning the model to the specific data; 2) customisable kernels that allow implementing non-linear cost models (function \( K \) in equation 1.1); 3) the ability to learn the cost model from any sets of manually-aligned video pairs; 4) the use of a maximum-margin training objective that has a proven reputation for accurate prediction. Rather than a distance, equation 1.1 expresses a complementary “similarity” which can be equally used for the aim of classification. In the equation, \( s \) and \( t \) represent the two sequences, \( y \) is the alignment path, \( \psi(s, t, y) \) is a mapping extracted from them, and \( w \) are the model’s parameters. Further details will be provided in the later chapters.

\[
\begin{align*}
w^T \psi(s, t, y) &= \sum_{k=1}^{[n]} w_{yk-1,yk}^{tr} + w_{em}^T K(s_i - t_j) I[y_k = M] \\
w_{0,s}^{tr} &= 0; \quad I[y_k = M]: i++, j++] \\
I[y_k = S]: j++; \quad I[y_k = T]: i++
\end{align*}
\] (1.1)

In the rest of this thesis we will explore these two different classification approaches, both leveraging the EHMM-SSVM, over several experiments. As minimum-distance classifier, we will adopt the \( k \)-nearest neighbours classifier (\( k \)-NN) [21]. To curb its computational complexity, we will apply a prototype selection technique that abates the overall number of comparisons. As feature-based classifier, we will insted adopt a multi-class SVM.
1.4 Contributions

In this thesis, we present an alignment model (EHMM-SSVM) and a distance function computed with the EHMM-SSVM. The alignment model is based on an extended hidden Markov model and the framework of regularized risk minimization (i.e., the structural support vector machine [15]). We also investigate a two dissimilarity-based approaches for classification. Our main contributions include:

- proposing an alignment algorithm integrating an extended (or paired) HMM with the structural SVM, and providing two dedicated loss functions (Chapter 3);
- improving the existing paired HMM by removing restrictions to the transition matrix (extended HMM), and discussing various similarity functions for an improved alignment (Chapter 4);
- proposing an efficient distance function based on the EHMM-SSVM for the classification of human actions (Chapter 5).

Overall, these contributions form an innovative approach to the alignment and classification of action videos and appear sizeable for a doctoral dissertation.

1.5 Publications

1.6 Thesis Outline

The thesis is organized into six chapters that are summarized as follows:

Chapter 2. This chapter presents the background knowledge for the proposed contributions. It discusses the concepts of different alignment algorithms, and the learning of minimum-risk (aka maximum-margin) classifiers. The structured prediction and pairwise sequence alignments are the focus of the entire discussion and are the foundation of Chapters 3 and 4. Feature-based classifiers are also introduced for the benefit of Chapter 5.

Chapter 3. This chapter discusses a novel alignment algorithm - named pair hidden Markov support vector machine (PHMM-SSVM) - that combines the features of a PHMM with the effective parameter estimation of the structural support vector machine (SSVM). It uses a maximum-margin training objective for the transition matrix of the pair HMM. The experiment has tested it against state-of-art alignment algorithms and achieved remarkable alignment accuracy.

Chapter 4. In Chapter 4, the pair hidden markov model (PHMM) is extended to the EHMM by removing certain restrictions on the transition matrix. This extension includes an extended hidden model and different (dis)similarity functions. The discussion indicates that the alignment performance can be tuned by choosing the (dis)similarity function based on the dataset characteristics.

Chapter 5. The challenge of dissimilarity-based classification is tackled in terms of minimum distance from templates. In this chapter, we propose leveraging the extended hidden Markov support vector machine (EHMM-SSVM) proposed in Chapter 4 for measuring distances between pairs of sequences. We have applied our method with two classifiers: nearest neighbours and DAGSVM. The experimental results over two popular action datasets have shown that our approach has been capable of achieving an accuracy higher than many existing methods and comparable to a state-of-the-art algorithm.

Chapter 6. We have concluded our discussion with a summary of the main findings and suggestions for possible future extensions.
Chapter 2

Review of Related Works

The aim of this chapter is to offer a review of the literature required to understand the contributions of this thesis. We will first review different sequence alignment algorithms for finding optimal alignment paths between two time-stamped sequences. The two main families of algorithms are Dynamic Time Warping (DTW) and modified hidden Markov models (HMMs). An overview of DTW and its most current, state-of-the-art variant - Generalized Canonical Time Warping (GCTW) - will be provided as term of comparison. Following that, we will review the hidden Markov model and its paired variant. Such a pair hidden Markov model is used in conjunction with structural SVM, a performing approach for automatically learning the model’s parameters from a set of manually-aligned sequence pairs. This approach allows for minimum-risk (aka maximum-margin) training of the model’s parameters and has a proven reputation for the accurate prediction on structured data. Lastly, since the optimal alignment of paired sequences can behave as a distance function between them, we have then focused on the application of our model to different classifiers such as k-NN and SVM. Therefore, they are also reviewed in this chapter. Another section overviews prototype selection, which is an approach for mitigating the high storage requirements, slow response and low noise tolerance of the k-NN classifier. Eventually, the chapter is concluded by a brief literature review of applications of sequence alignment to the recognition of human actions in the field of computer vision.
2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a well-known algorithm for finding an optimal alignment and a similarity measure between two given time series. The optimal alignment is computed by a nonlinear mapping of the frame indexes from the two sequences under certain restrictions. Because of its ability to align speech utterances of different execution rate, DTW was originally proposed for speech recognition, to match a sampled voice with others of different sample rates. DTW has also been applied to temporal sequences of frames (i.e., videos), and any data which can be turned into a sequence. For instance, given two human walking sequences, \( s = (s_1, s_2, ..., s_N) \) of length \( N \) and \( t = (t_1, t_2, ..., t_M) \) of length \( M \), with different intrinsic speed, DTW finds an alignment from a series of local cost estimates, \( c(s_i, t_j) \), which are low if \( s_i, t_j \) are similar or high if otherwise. By computing all possible local costs for sequences \( s \) and \( t \), a cost matrix can be obtained as \( C(N, M) \) where \( C \in \mathbb{R}^{N \times M} \). Then, the optimal alignment of the two walking sequences is the alignment with minimal overall cost, obtained along the path with most similar pairs under certain constraints. In practice, even if one person was walking faster than the other, or if there were accelerations and decelerations during the observation, the optimal alignment can be determined.

In general, the following definition [22] formalises the notion of an alignment:

**Definition 2.1.** An \((N, M)\)-warping path (or simply referred to as warping path if \( N \) and \( M \) are clear from the context) is a sequence \( p = (p_1, ..., p_L) \) with \( p_l = (n_l, m_l) \in [1 : N] \times [1 : M] \) for \( l \in [1 : L] \) satisfying the following three conditions.

(i) Boundary condition: \( p_1 = (1, 1) \) and \( p_L = (N, M) \)

(ii) Monotonicity condition: \( n_1 \leq n_2 \leq \ldots \leq n_L \) and \( m_1 \leq m_2 \leq \ldots \leq m_L \)

(iii) Step size condition: \( p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\} \) for \( l \in [1 : L - 1] \)

The boundary condition ensures that the initial elements and the last elements from two sequences are aligned to each other. The monotonicity condition and step size condition prevent the operations of alignment to go backwards, ensure that all elements are involved, and prevent replications. Note that Definition 2.1 only states that a warping path (NOT
Chapter 2. Review of Related Works

Figure 2.1: Illustration of paths of index pairs for some sequence $s$ of length $N = 9$ and some sequence $t$ of length $M = 7$. (A) Admissible warping path satisfying the conditions (i), (ii), and (iii) of Definition 2.1. (B) Boundary condition (i) is violated. (C) Monotonicity condition (ii) is violated. (D) Step size condition (iii) is violated.

an optimal warping path) contains a sequence of non-repeating pairs of elements under these three conditions. The number of admissible warping paths between two sequences is exponential. Figure 2.1 illustrates an example of a correct warping path and three wrong warping paths under the three conditions.

To find the optimal warping path $p^*$ with minimal total cost, one could test every admissible warping path between $s$ and $t$. However, this is a massive computation procedure that would lead to an exponential complexity in the lengths $N$ and $M$. Dynamic Programming can solve this problem by a sophisticated algorithm that achieves only a quadratic cost, $O(NM)$. The algorithm can be described in terms of an accumulated cost matrix, $D(n, m)$, which allows finding the optimal path, $p^*$, in reverse order from the indices of the last cost cell, $p_L = (N, M)$. The next definition from [22] shows how the accumulated cost matrix $D$ can be computed efficiently.

**Definition 2.2.** The accumulated cost matrix, $D$, can be computed iteratively based on the following properties:

(i) $D(n, 1) = \sum_{k=1}^{n} c(s_k, t_1)$ for $n \in [1 : N]$

(ii) $D(1, m) = \sum_{k=1}^{m} c(s_1, t_k)$ for $m \in [1 : M]$

(iii) $D(n, m) = \min \{ D(n-1, m-1), D(n-1, m), D(n, m-1) \} + c(s_n, t_m)$ for $1 < n \leq N$ and $1 < m \leq M$

After the accumulated cost matrix $D$ is computed, the following algorithm from 1 can be used to determine the optimal warping path, $p^*$. 

Algorithm 1: Optimal warping path: main steps.

**Input**: Accumulated cost matrix $D$

**Output**: Optimal warping path $p^*$

**Procedure**: The optimal path $p = (p_1, ..., p_L)$ is computed in reverse index order starting from $p_L = (N, M)$. Assume $p_l = (n, m)$ has been computed. In case $(n, m) = (1, 1)$, we must have $l = 1$ and the algorithm is finished. Otherwise,

$$p_{l-1} = \begin{cases} (1, m - 1), & \text{if } n = 1 \\ (n - 1, 1), & \text{if } m = 1 \\ \arg\min \left\{ D(n - 1, m - 1), D(n - 1, m), D(n, m - 1) \right\} & \text{otherwise} \end{cases}$$

(2.1)

where we take the lexicographically smallest pair in case “argmin” is not unique.

Figure 2.2 shows a visual example of cost matrices and optimal warping path for two hypothetical sequences. To better explain the algorithm, in the following we also present a small numerical example in detail. Given two sequences $s = \{1, 2, 3, 5, 5, 6\}$ of length $N = 7$ and $t = \{1, 1, 3, 4, 3, 5\}$ of length $M = 6$, the cost matrix $C$ has been obtained via Definition 2.1 and is shown in table 2.1a. We can then easily find the the accumulated cost matrix $D$ based on the cost matrix $C$ via Algorithm (iii): $D(n, m) = \min \{D(n - 1, m - 1), D(n - 1, m), D(n, m - 1)\} + c(s_n, t_m)$ from Definition 2.2. Table 2.1b shows the accumulated cost matrix $D$. 
Chapter 2. Review of Related Works

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (value)</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

(A) Cost matrix between s and t

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (value)</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>11</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

(B) Accumulated cost matrix between s and t

Table 2.1: Numerical example of the cost matrix and the accumulated cost matrix determined during the formation of the optimal warping path. (A) Each cell represents the distance between elements $s_i$ and $t_j$ and is computed as cost function $C(s_i, t_j) = |s_i - t_j|$. (B) The accumulated cost matrix is obtained using Definition 2.2.

As one can see in table 2.1b, we have iterated from the end pair of elements to the first pair of elements picking the minimum value at each step. The path formation starts at $D(6, 5) = 3$ (bottom-right) and ends at $D(1, 1) = 0$ (top-left), and follows continuously restricted steps for choosing the minimum score from its three neighbours $D(n-1, m-1), D(n-1, m), D(n, m-1)$. The yellow highlights show the optimal warping path, $p^*(s, t) = \{(1, 1), (2, 2), (3, 3), (3, 4), (3, 5), (4, 6), (5, 6), (6, 6), (7, 6)\}$.

2.2 Generalized Canonical Time Warping

Time series are a ubiquitous form of data occurring in virtually every field. The temporal alignment of time series is therefore an urgent problem for many scientific disciplines. Dynamic time warping (DTW) is a technique for efficiently achieving this alignment in tasks as diverse as gesture recognition, robotics, speech processing, etc. Although DTW has been successfully used in many types of research, the alignment of time series describing human actions poses two fundamental challenges: 1) the alignment in large, multidimensional spaces and the consequent need for implementing feature selection; and 2) a large variation in the motion style and the speed in performing activities. Generalized canonical time warping (GCTW) is a state-of-the-art spatial alignment method which maximises the correlation between two multi-dimensional time series to find their optimal alignment.
GCTW extends DTW by leveraging canonical correlation analysis (CCA) to measure the spatial correlation and introduces a feature-weighting mechanism. GCTW is able to align signals of different dimensionality and provide higher weights to the features that make both signals more correlated.

### 2.2.1 Canonical Correlation Analysis (CCA)

Canonical correlation analysis (CCA) is a multivariate statistical model that facilitates the study of inter-relationships among two sets, X and Y, of multiple variables. The problem can be defined as finding two sets of basis vectors (one for X and the other for Y) such that the correlations between the linear combination of X and Y onto the basis vectors are mutually maximised. The purpose of CCA is to explain the nature of whatever relationships exist between X and Y via the maximal correlation of linear combinations rather than the original correlation between X and Y, $\text{Corr}(X, Y)$. This is because the dimension of matrix

\[
\text{Cov}(X) = \Sigma_{xx}; \quad \text{Cov}(Y) = \Sigma_{yy}; \quad \text{Cov}(X, Y) = \Sigma_{xy} = \Sigma_{yx}^T
\]

impacts the computational complexity dramatically. In addition, similarly to principal component analysis (PCA), CCA can provide a beneficial noise reduction. The advantage of CCA over PCA (of the two sets separately) is its ability to perform the analysis of X and Y jointly.

To reduce the computation complexity, CCA focuses on the correlation between a linear combination of the variables in the set X and a linear combination of the variables in the other set Y. The canonical variables are the pairs of linear combinations, which have the largest correlation among all pairs that are uncorrelated with respect to the previously selected pairs. Assuming there are two sets of variables ($X \in \mathbb{R}^{d_x \times m}$) and ($Y \in \mathbb{R}^{d_y \times n}$) where $d_x$ and $d_y$ are the dimensions of two sets and $m$ and $n$ ($m \geq n$) are the length. CCA seeks vectors $a$ and $b$ that maximise the correlation $\rho$. 
Chapter 2. Review of Related Works

\[ \text{Cov}(U) = \text{Cov}(a^T X) = a^T \Sigma_{XX} a; \quad \text{Cov}(V) = \text{Cov}(b^T Y) = b^T \Sigma_{YY} b \]

\[ \text{Cov}(U, V) = \text{Cov}(a^T X, b^T Y) = a^T \Sigma_{XY} b \]

\[ \rho = \text{corr}(a^T X, b^T Y) = \frac{a^T \Sigma_{XY} b}{\sqrt{a^T \Sigma_{XX} a \cdot b^T \Sigma_{YY} b}} \]

**Definition 2.3.** Fibration 1st Pair of Canonical Variables \((U_1 \text{ and } V_1)\) have unit variances which maximise the correlation \(\rho_1\) [23].

Derivation of 1st Pair of Canonical Variables \(\rho_1 = \text{corr}(a_1^T X, b_1^T Y) = \frac{a_1^T \Sigma_{XY} b_1}{\sqrt{a_1^T \Sigma_{XX} a_1 \cdot b_1^T \Sigma_{YY} b_1}}\). By the Cauchy-Schwarz inequality, it can be derived:

\[ c_1^T \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2} d_1 \leq (c_1^T \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2} \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{XX}^{-1/2} c_1)^{1/2} (d_1^T d_1)^{1/2} \]

\[ \Rightarrow c_1^T \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2} \Sigma_{XX}^{-1/2} c_1 \leq \rho_1^2 c_1^T c_1 \]

\[ \Rightarrow \text{Corr}(a_1^T X, b_1^T Y) = \frac{\rho_1^2 c_1^T c_1 \cdot \sqrt{d_1^T d_1}}{\sqrt{c_1^T c_1} \cdot \sqrt{d_1^T d_1}} = \rho_1 \]

\[ \Rightarrow c_1 = \Sigma_{XX}^{1/2} a_1; \quad d_1 = \Sigma_{YY}^{-1/2} \Sigma_{XY} \Sigma_{XX}^{-1/2} c_1 \]

1st Pair of Canonical Variables: \(U_1 = c_1^T \Sigma_{XX}^{-1/2} X, V_1 = d_1^T \Sigma_{YY}^{-1/2} Y\)

**Definition 2.4.** Fibration 2nd Pair of Canonical Variables \((U_2 \text{ and } V_2)\) have unit variances which maximise the correlation \(\rho_2\) among all choices that are uncorrelated with 1st pair of canonical variables [23].

**Definition 2.5.** Fibration k-th Pair of Canonical Variables \((U_k \text{ and } V_k)\) have unit variances which maximise the correlation \(\rho_k\) among all choices that are uncorrelated with the previous \(k - 1\) pairs of canonical variables [23].
Chapter 2. Review of Related Works

According to the last three definitions, CCA enjoys the following properties for \( k \neq l \):

\[
\begin{align*}
\text{Var}(U_k) &= \text{Var}(V_l) = 1 \\
\text{Cov}(U_k, V_l) &= \text{Corr}(U_k, V_l) = 0 \\
\text{Cov}(U_k, U_l) &= \text{Corr}(U_k, U_l) = 0 \\
\text{Cov}(V_k, V_l) &= \text{Corr}(V_k, V_l) = 0
\end{align*}
\]

(2.4)

2.2.2 Combination of CCA and DTW

Canonical time warping [12] extends DTW by adding a feature selection mechanism via measuring CCA correlation between the two sequences. Feature selection is a process of selecting a subset of important features to increase the accuracy of the alignment. CCA minimises the distances between the two sequences by finding the linear combinations of the variables in \( X \) that correlate mostly with the linear combinations of the variables in \( Y \). In [12], the distance function was defined as:

\[
D_{\text{cca}}(X, Y) = \left\| a_x^T X - b_y^T Y \right\|^2 \quad \text{s.t.} \quad a_x^T XX^T a_x = b_y^T YY^T b_y = I_k
\]

(2.5)

where \( k \) denotes dimensional embeddings that preserve most of the energy of the pair of canonical variables \((a_x^T X, b_y^T Y)\). According to the CCA properties, the pair of canonical variables that achieves the maximum correlation must be orthogonal to the other pairs. In order to find the canonical variables, \( a \) and \( b \) can be determined from a generalised eigenvalue problem:

\[
a = \begin{bmatrix} 0 & XY^T \\ YX^T & 0 \end{bmatrix}, \quad b = (1 - \lambda) \begin{bmatrix} XX^T & 0 \\ 0 & YY^T \end{bmatrix} + \lambda I.
\]

At its turn, the optimal warping path minimising the sum of the aligned distances of \( X \) and \( Y \) can be written as:

\[
D_{\text{dtw}}(X, Y) = \sum_{t=1}^{k} \left\| X_{p_t^X} - Y_{p_t^Y} \right\|^2
\]

(2.6)
where \( k \) is the length of the optimal warping path \( p = (p_x, p_y) \) that is parametrised by pairs of vectors from \( X, Y \) respectively.

We can see that the distance functions of \( D_{ca} \) and \( D_{dtw} \) have similar expressions. To join CCA and DTW, Zhou [12] has introduced the matrices \( W_x \) and \( W_y \) to encode the warping path in matrix form and has rewritten 2.6 as:

\[
D_{dtw}(X, Y, W_x, W_y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} w_{ij} x_i w_{ij}^T \| x_i - y_j \|^2 = \| X W_x^T - Y W_y^T \|^2 \tag{2.7}
\]

where \( W_x \in \{0,1\}^{m, n_x}, W_y \in \{0,1\}^{m, n_y} \) encode binary selection matrices that indicate the alignment path.

Eventually, GCTW [12] combines DTW and CCA by minimising:

\[
D_{ctw}(a_x, b_y, W_x, W_y) = \| a_x^T X W_x^T - b_y^T Y W_y^T \|^2 \tag{2.8}
\]

According to the CCA properties, GCTW must abide by the following constraints:

(i) \( X W_x^T 1_m = 0, Y W_y^T 1_m = 0 \)
(ii) \( a_x^T X W_x^T W_x X^T a_x = b_y^T Y W_y^T W_y Y^T b_y = I_b \)
(iii) \( a_x^T X W_x^T W_y Y^T b_y \) to be a diagonal matrix

**Algorithm 2:** Generalized Canonical Time Warping: main steps.

**Input:** \( X, Y \)

**Output:** \( a_x, b_y, W_x, W_y \)

Initialize \( a_x = I_x, b_y = I_y, k \) for energy reservation

**repeat**

- Use dynamic programming to compute optimal warping path, \( W_x, W_y \)
- Compute canonical variables \( a_x^T X, b_y^T Y \)
- as the leading \( k \) generalized eigenvectors of:

\[
\begin{bmatrix}
0 & X W_x^T W_y Y^T \\
Y W_y^T W_x X^T & 0
\end{bmatrix}
\begin{bmatrix}
a_x^T \\
b_y^T
\end{bmatrix}
= \begin{bmatrix}
X W_x^T W_x X^T & 0 \\
0 & Y W_y^T W_y Y^T
\end{bmatrix}
\]

**until** \( D_{ctw} \) converges.

Chapter 2. Review of Related Works

2.3 Hidden Markov model

In the classification literature, the term “generative model” is used to refer to a joint distribution over both the observations (aka measurements) and the classes expressed as the product of a likelihood and a prior. Such models have several appealing features: in case new classes need to be learned, their likelihoods can be derived independently from those of all the existing classes. In addition, the new priors can be obtained with minimal adjustments. A generative model also allows for the determination of the marginal density of an observation, useful, for instance, for distinguishing outliers of all classes and generally for prediction purposes. Generative models include popular classifiers such as Naive Bayes and the hidden Markov model.

The hidden Markov model can be first introduced with reference to an unsupervised learning scenario. In this case, we refer to the “state” of a system that cannot be directly observed, yet inferred through measurements of other variables, or observations. This implies that the state of the 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) is a generative model consisting of
Chapter 2. Review of Related Works

a set of hidden states \( z_{1..T} \) and observations \( y_{1..T} \), joint in a graphical model as shown in figure 2.3 (please note that we use symbols \( z \) for the states and \( y \) for the measurements in the following to adhere to the notations used most commonly for an HMM). An HMM is based on two fundamental independence assumptions [24]:

(i) Transition from a state to the next one is only dependent on the current state, not the previous ones.  
\[
p(z_t|z_1, ..., z_{t1}, y_1, ..., y_{t1}) = p(z_t|z_{t1})
\]

(ii) Each observation is only dependent on its current state, not any other observation or state. 
\[
p(y_t|z_1, ..., z_T, y_1, ..., y_T) = p(y_t|z_t)
\]

An HMM is fully defined by the factorised joint probability of \( z_{1..T} \) and \( y_{1..T} \):

\[
p(z_{1..T}, y_{1..T}) = p(z_1) \sum_{t=2}^{T} p(z_t|z_{t1}) \sum_{t=1}^{T} p(y_t|z_t)
\]  \hspace{1cm} (2.9)

Since the \( z_t \) are categorical variables, this model requires the probabilities of the initial state, \( z_1 \), and a matrix of transition probabilities from the previous to the current state value. In addition, it requires a likelihood for measurement \( y \) for every possible state value. While the definition of the model is relatively straightforward, there are several, non-trivial problems attached to an HMM:

(i) computing the probability of a given sequence of measurements \( z_{1..T} \). This is a marginal of (2.9) and can be obtained with the forward-backward algorithm;

(ii) inferring the most probable sequence of states from a given sequence of measurements. This is obtained with the Viterbi algorithm;

(iii) training the model in an unsupervised way, i.e., based on a training set with only sequences of measurements and no knowledge of the corresponding states. This is obtained with the Baum-Welch algorithm, an instance of expectation-maximisation;

(iv) training the model in a supervised way, i.e., based on a training set with both sequences of measurements and the corresponding manually-annotated sequences of states. This problem boils down to the separate training of all the factors in (2.9) and is the case relevant to this thesis.
In particular, the famed Viterbi algorithm solves the problem of estimating the state sequence of an HMM. It is a dynamic programming variant that finds a global optimum from a succession of local decisions. Assuming \( N \) possible values for the state, and indexes \( i, j \in 1..N \), we can give a formal definition of the Viterbi recursion as follows:

**Initialization:**

\[
v_1(j) = p(z_j|z_0)p(y_1|z_j)
\]

**Recurrence:**

\[
v_t(j) = \max_{i=1}^{N} v_{t-1}(i)p(z_j|z_i)p(y_t|z_j); 1 < j \leq N, 1 < t \leq T
\]

**Termination:**

The best score:

\[
P^* = v_T = \max_{i=1}^{N} v_T(i)
\]

The start of backtrace:

\[
Q^* = b_T = \arg\max_{i=1}^{N} v_T(i) \cdot p(z_T|z_i)
\]

### 2.4 Pair hidden Markov model

The HMM can be easily extended to formally accommodate the problem of alignment by using two sequences of measurements instead of one, and by interpreting the sequence of states as the warping path. This extension has been referred to in the literature as “pair HMM”, or PHMM, a probabilistic model for pairwise sequence alignments. Given two sequences, \( s = \{s_1, ..., s_i, ..., s_L_s\} \) and \( t = \{t_1, ..., t_j, ..., t_L_t\} \), their alignment can be intuitively defined as a sequence of index pairs from the two sequences. However, to simplify both notations and operations, the alignment is re-defined as a sequence of only three types of symbols: \( M \) (“match”), \( S \) (“insert a gap on sequence \( s \)”) and \( T \) (“insert a
gap on sequence \( t' \)). The symbols have the following meaning: assuming \( i \) and \( j \) to be the current indices over sequences \( s \) and \( t \), respectively, 1) symbol \( M \) pairs frames \( s_i \) and \( t_j \) and then increments both indices; 2) symbol \( S \) pairs no frames and only increments index \( j \); and, likewise, 3) symbol \( T \) pairs no frames and only increments index \( i \). As a toy example, we show below a possible alignment for two short sequences from character set \( \{A, B, C, D\} \):

\[
\begin{align*}
s &= A \ B \ C \ B \ D \ A \ D \\
t &= A \ B \ D \ C \ B \ A \ D \\
y &= M \ M \ S \ M \ M \ T \ M \ M
\end{align*}
\] (2.16)

In the above example, sequence \( y \) encodes the alignment, with the \( M \) symbols showing the matched frames (e.g, \( s_3 \) and \( t_4 \)) and the \( S \) and \( T \) symbols accounting for the required gaps. The length of the alignment is bounded between \( \max(L_s, L_t) \) and \( L_s + L_t \).

In probability notation, a PHMM is a model for the joint probability, \( p(s, t, y) \), of the two sequence, \( s \) and \( t \), and their alignment, \( y \). Such a model can be used to infer an optimal alignment, \( \bar{y} \), for the two sequences as \( \bar{y} = \arg\max_y p(s, t, y) \). Like for a conventional HMM, the joint probability of a PHMM factorises into a set of transition and emission probabilities. In the original PHMM, the transition matrix has been simplified to follow intuitive constraints, with the transition probabilities being defined as: (1) \( \delta \) for transitions from \( M \) to either \( S \) or \( T \); (2) \( \varepsilon \) for staying in \( S \) or \( T \); (3) \( 1 - \varepsilon \) for transitions from either \( S \) or \( T \) to \( M \). Note that this model bars direct transitions from \( S \) to \( T \) and vice versa assuming that a pair of matched frames will always follow a run of gaps. Figure 2.4 shows the state diagram of the PHMM, while Table 2.2 shows the complete transition probabilities table.

To complete the model, we also need to define the emission probabilities. To this aim, we note the probability of emitting aligned pair \((a, b)\) as \( p_{a,b} \) and the probability of emitting measurement \( a \) against a gap as \( q_a \). In the common case of numerical measurements, both \( p \) and \( q \) will be multivariate likelihoods such as Gaussian distributions or mixture models.

Using a PHMM, the optimal alignment for a pair of sequences can be found via an equivalent Viterbi algorithm [25]. Its computational complexity, \( O(L_sL_t) \), is only linear in the
Chapter 2. Review of Related Works

length of each sequence (and quadratic overall) thus ensuring reasonably fast alignments.

The main steps of the algorithm are given below, where the probability of reaching state

\( * = \{M, S, T\} \) at indices \( i \) and \( j \) over \( s \) and \( t \) is noted as \( p^*(i, j) \).

**Initialization:** \( p^M(1, 1) = p^S(1, 1) = p^T(1, 1) = p^*(0, j) = p^*(i, 0) = 1. \)

**Recurrence:** \( i = 1, ..., L_s, j = 1, ..., L_t: \)

\[
\begin{align*}
p^M(i, j) &= p_{s,t} \max \left\{ (1 - 2\delta) p^M(i-1, j-1), (1 - \varepsilon) p^S(i-1, j-1), (1 - \varepsilon) p^T(i-1, j-1) \right\} \\
p^S(i, j) &= q_{s,t} \max \left\{ \delta p^M(i-1, j), \varepsilon p^S(i-1, j) \right\} \\
p^T(i, j) &= q_{s,t} \max \left\{ \delta p^M(i, j-1), \varepsilon p^T(i, j-1) \right\}
\end{align*}
\]

**Termination:**

\[
p(s, t, y) = \max(p^M(L_s, L_t), p^S(L_s, L_t), p^T(L_s, L_t))
\]
2.5 Prototype Selection

Prototype selection is a method often used in classification to choose a representative subset of samples from each class. The selected prototypes are then used either with minimum-distance classifiers, or to form a dissimilarity-based feature vector for feature-based classification. Such a representative set should be selected as the subset of prototypes which can yield a similar contribution to the classification as the whole initial set. Indeed, the main drawback of nonparametric distance-based classifiers such as $k$-NN is their computational complexity at runtime: in principle, each test sample should be compared with every sample in the training set. While mitigation techniques such as the use of the triangle inequality and k-d trees can be exploited, the problem remains intrinsically complex, especially for large datasets. An alternative to the full run-time search is offered by prototype selection: in this case, the training set is replaced by a subset of representative prototypes, making the search substantially faster. In addition to reducing the run-time complexity, prototype selection typically achieves a comparable or even higher classification accuracy than the full search thanks to the removal of noisy and redundant samples [26]. The choice of the best prototypes can be performed according to many different criteria, including uniform distribution, centrality in the class, and others [17].

The followings selection techniques are amongst the most popular for selecting a prototype set: Random, RandomC, KCentres, LinProg, Modesseek, FeatSel, Border prototype selector, and EdiCon. They will be briefly described hereafter using the following notations: $m$ classes, $(c_1, ..., c_m)$; $T$, the training set; $T_{c_i}$, the training samples from class $c_i$.

**Random**: This method selects $P$ prototypes randomly from the training set $T$ [27]. It can work well for large prototype sets since neighbouring objects are similar prototypes candidates. However, the random selection may be less successful if a small prototype set is needed. Besides, as a disadvantage, it is possible to find redundant prototypes with this method.

**RandomC**: It is similar to Random but it selects $K$ objects per class and stacks together $P = mK$. 
**KCentres**: It is an unsupervised method based on clustering non-numerical objects. Given a parameter $K$ for the number of objects from each class, this method tries to obtain $K$ objects from each class $c_i$. These selected objects from the related class are evenly distributed with respect to the notion of dissimilarity used for the clustering. For each class, *KCentres* chooses $K$ prototypes with these steps [27]:

1. randomly pick an initial set of prototypes, $P = \{p_1, \ldots, p_K\}$;
2. partition all the class’ samples into $K$ subsets, $J_1 = \{p_1, \ldots, p_K\}$, based on their closest prototype;
3. for each $J_l, l = 1, 2, \ldots, K$, find its most central element (the element whose maximum distance to all other elements is minimum);
4. replace the prototypes in $P$ with the most central elements.

**LinProg**: It is a supervised method which trains a sparse separating hyperplane $f(D(x, P)) = w^T D(x, P) + b = \sum_{i=1}^{n} w_i d(x, p_i) + b$ in a dissimilarity space $D(T, P)$. This linear function is obtained by solving a linear programming problem, where a sparse solution is to minimise the weight vector $w$. $w_i$ are expressed by non-negative variables $\alpha_i$ and $\beta_i$ as $w_i = \alpha_i - \beta_i$. In the optimization problem it is also introduced a non-negative slack variable $\xi_i$ that accounts for classification errors as well as a regularization parameter $\gamma$. For a set of training objects $x_i \in T$ with class labels $y_i \in (1, -1)$, the minimization problem is formulated as follows:

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} (\alpha_i + \beta_i) + \gamma \sum_{i=1}^{n} \xi_i \\
\text{s.t.} & \quad y_i f(D(x_i, P)) \geq 1 - \xi_i, \; i = 1, \ldots, n \\
& \quad \alpha_i, \beta_i, \xi_i \geq 0
\end{align*}
\]  

(2.21)

The final prototypes are the objects that have $w_i$ weights different from 0. The authors state that this procedure may be beneficial for two-class problems from a computational
point of view, but for multi-class problems not so much since it may result in a large set of prototypes.

**Modeseek:** This method is focusing on the modes in the dissimilarity data by estimating each object \(x_i\) with its nearest neighbours \(x_{n_i}\). The selected prototypes are the object set which have the optimized minimum dissimilarity to their neighbours \(x_{n_i}\). The algorithm in a class-wise way proceeds as follows [27]:

1. Set a relative neighbourhood size as an integer \(n > 1\).
2. For each object \(x_i \in T_{c_i}\) find the dissimilarity \(d_{nn} = d(x_i, x_{n_i})\) to its i-th neighbour.
3. Find a set \(p_{m_i}\), consisting of all objects from training set \(T_{c_i}\) for which \(d_{nn}\) is minimum within its set of \(n\) neighbours.

**FeatSel:** It is a supervised greedy forward selection optimized for dissimilarity data. It uses the forward feature selection [28] and the leave-one-out (LOO) 1-NN error as a selection criterion. This method is very fast since it operates comparisons and sorts directly in the dissimilarities space [27]:

\[
\begin{align*}
\min D &= \sum_{x_i \in T, x_i \notin P} \text{LOO}(x_i) \\
\text{LOO}(x_i) &= \begin{cases} 
1 & \lambda_T(x_i) \neq \lambda_P(p') \\
0 & \lambda_T(x_i) = \lambda_P(p')
\end{cases} \\
p' &= \arg\min_{p_j \in P} d(x_i, p_j)
\end{align*}
\]

(2.22)

which \(\lambda_T(x)\) is the class label of \(x \in T\), \(\lambda_P(p')\) is the class label of \(p \in P\) and \(D\) is the 1-NN classification error of the training set \(T\) classified by the prototypes set \(P\).

**EdiCon:** This is an editing and condensing algorithm applied to an entire dissimilarity representation. The editing method removes those objects that are erroneously classified by the 1-NN, so the overlapping of classes is decreased, and condensing improves the performance of the 1-NN classifier on the new set.
Border prototype selector: As its name suggests, this method is based on selecting those objects belonging to the border of the data distribution [29]. It is important to emphasize that whether one object belongs to the center or border of the dissimilarity data is determined by the dissimilarity measure used and the problem at hand. Especially for dissimilarities computed on top of complicated structures such as graphs, a center or border object may be tricky to define. In this method, the problem of the similar contribution of the center prototypes selected by the previous method is avoided. On the other hand, outliers are likely to be selected, since they usually are around the borders of the data distribution.

Spanning prototype selection: Starting from the set median string, the next prototype to be added to the representative set R is the object with largest minimum distance to the current set of prototypes [29]. Formally, this set is that fulfilling:

$$\text{argmax}_{x_i \in T, p_j \in P} [\min d(x_i, p_j)]$$  \hspace{1cm} (2.23)

where $T$ is training set and $P$ are the prototypes selected so far.

In this way, objects that yield similar contribution to the representative set are not likely to be selected, since the next prototype is always the farthest one from the already selected set. Also, the prototypes have a tendency to be uniformly distributed. The authors of [29] also point out that outliers are likely to be selected since they have a large distance from other objects. Actually, this method is a variant of the farthest first transversal (FFT) [30] which was originally proposed for KCenters clustering initialization [31]. The difference between the two is that the FFT starts from a random object and not from the median object.

2.6 Support vector machines

Support vector machines (SVM) are a well-known supervised learning model for classification. Since circa the nineties, they have established a solid reputation for accuracy and efficiency in all fields of science. One of the much-cited advantages of SVM is that it can use kernels, an efficient way to embed non-linearities in the classifier. Another significant
advantage of SVM is that its training procedure guarantees a global optimum because the
determination of the model parameters corresponds to a convex optimisation problem. In
this section, we will review the original binary SVM and its soft-margin extension.

2.6.1 Binary SVM

Vladimir Vapnik is the celebrated author of the binary SVM and of many of its extensions[32].
The essential aim of binary SVM is to construct a separating surface between two classes
that minimises an upper bound on the out-of-sample (or generalisation) error. This is
achieved by simultaneously minimising the empirical risk associated with the model and a
bound over the generalisation error. In the case of a linear plane separating the elements
of the two classes, SVM selects the plane that maximises the distance from the closest
elements of each class. For instance, figure 2.5 shows a line $y(x_n) = w^T \phi(x) + b = 0$ which
separates two classes (C1 and C2) where $y(x_n) > 0$ for points located on class C1 and
$y(x_n) < 0$ for class C2. We call this separating line a “decision boundary” or “hyperplane”.
The hyperplane $y(x_n)$ can change its position and orientation by varying its parameters
$(w, b)$. As we can see, to classify the points confidently, SVM has to find the hyperplane
$w, b$ that keeps maximal distance from the closest points of both classes. Such points are
known as “support vectors” (the two red dots and the three green triangles in the figure).

The goal of SVM training is to find the parameters of separating hyperplane $(w, b)$ that
maximizes the geometric margin. Let us assume samples $(x_i, y_i), i = 1, ..., n$, where $x_i$
Chapter 2. Review of Related Works

means a feature vector and $y_i$ a label in (C1, C2). The margin of samples to the hyperplane is defined as: $y_i(w^T \phi(x) + b) > 0$, where $\phi(x)$ denotes a feature space and $b$ is the bias parameter. For a new sample $x_j$, if $w^T \phi(x_j) + b > 0$, then $x_j$ is in class C1 (or $y_j = +1$). In contrast, if $w^T \phi(x_j) + b < 0$ means $x_j$ is in class C2 (or $y_j = -1$). Furthermore, there is one more plane at each side of the hyperplane through the closest class elements (see figure 2.6). The geometric margin is defined as its distance from the decision boundary/hyperplane, which can be easily calculated as $2/\|w\|$. 

Figure 2.6: Geometric margins at each side of the hyperplane.

Figure 2.6 shows that the boundaries of the classes have geometric margins $1/\|w\|$. The maximum of the geometric margin implies the minimum of $\|w\|$. In the theory of statistical learning[32], such a geometric margin plays a fundamental role in the determination of a bound over the generalisation error, since the relationship between the geometric margin and misclassification is $\text{MisclassificationRate} \leq \frac{2R}{\delta}$, where $\delta$ means the distance between samples and the hyperplane and $R$ means the maximum value of the samples $R = \max \|x_i\|$. 

To find the hyperplane in figure 2.5 that maximises the distance between the closest points of the opposing classes, it is sufficient to minimise the norm of $w$ i.e. $\|w\|$. The objective function can be defined as:

$$\min \frac{1}{2} \|w\|^2$$

s.t. $\|w\| \neq 0$, $y_i(w^T \phi(x) + b) \geq 1, \forall i = 1, 2, ..., n$ (2.24)
Chapter 2. Review of Related Works

The objective function subject to the inequality constraints can be minimised in terms of a Lagrangian equation:

\[
L(w, b, \alpha_{1:N}) = \frac{1}{2} - \sum_{i=1}^{N} \alpha_i \left\{ y_i (w^T \phi(x_i) + b) - 1 \right\},
\]  

(2.25)

By differentiating the Lagrangian equation with respect to \( w \) and \( b \) and replacing the results in the equation, the Lagrangian becomes:

\[
\hat{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)
\]

s.t. \( \sum_{i=1}^{N} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1...N \)  

(2.26)

Now the dual problem can be solved by:

\[
\max_{\alpha_{1:N}} \left[ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \right]
\]

s.t. \( \sum_{i=1}^{N} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1...N \)  

(2.27)

The principled advantage of SVM for binary classification is the mentioned probabilistic bound over the generalisation error. However, its main practical advantage is its ability to classify linearly separable data or nonlinearly separable data. To obtain a nonlinear classifier, we can embed a kernel function into SVM to “virtually” map the inputs to a high dimensional feature space where linear classification can be performed, with a nonlinear effect in the original data space. To this aim, product \( \phi(x_i)^T \phi(x_j) \) in the dual formulation can be replaced by a kernel function. Common kernels are provided off-the-shelf, but it is also possible to specify custom kernels. The kernel function can be (but not limited to) any of the following: linear \( \langle x, x' \rangle \), polynomial \( (\gamma \langle x, x' \rangle + r)^d \), rbf \( \exp(-\gamma |x - x'|^2) \), etc.
2.6.2 Soft-margin SVM

In practical scenarios, it is rare that the given dataset is neatly separable in data space (Figure 2.5). Noise or inseparable data from the two classes make it impossible to separate them by a hyperplane perfectly, as shown in figure 2.7. In this case, a new set of variables ("slack" variables) \( \xi \) will be appended to the model for the penalty so that data points are allowed to be on the wrong side of the hyperplane [33]. The value of the slack variables will increase by the same amount as the missing required distance from the hyperplane.

![Figure 2.7: A single "outlier" can affect the separating hyperplane significantly.](image)

As we can see in figure 2.6, the margin is again set to 1. However, the slack variables provide a trade-off allowing the margin for particular samples to be less than 1, namely \( 1 - \xi_i \). With these modifications, our new primal problem for the so-called soft-margin SVM is posed as follows:

\[
\begin{align*}
\mathbf{w}^*, \mathbf{b}^* &= \arg\min_{\mathbf{w}, \mathbf{b}} \left( \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{N} \xi_i \right) \\
\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + \mathbf{b}) &\geq 1 - \xi_i, \xi_i \geq 0, i = 1...N
\end{align*}
\]

(2.28)

where \( C \) is a coefficient to weigh the cost of samples which do not meet the 1 margin requirement. \( C \) is set arbitrarily to balance the original objective (maximum-margin) and the penalty of the violating samples (minimum-risk).
The dual form of equation 2.28 can be solved in a similar way as equation 2.27:

$$\max_{\alpha_{1:N}} \left[ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \right]$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, i = 1...N$$

(2.29)

The only difference between equation 2.27 and equation 2.29 is the addition of a “box constraint”, $0 \leq \alpha_i \leq C$. Note that the “kernel trick” to build a nonlinear classifier can be applied also here.

### 2.7 Multi-Class SVM

Multi-class SVM is a support vector machine to classify more than two classes. As we have seen, binary SVM has the limitation of being a two-class classifier. There are two fundamental ways to tackle the multi-class extension: 1) building an extended objective that can train a multi-class SVM as a single objective; or 2) building a reduction that maps the multi-class problem to a number of binary SVM classifiers. The first approach is often coalesced with the classification of structured objects and is presented in the next subsection. The second approach - by far the most popular in the SVM literature - reduces the multi-class problem to multiple binary SVM problems in three intuitive ways: (i) a set of one-vs-all classifiers, one per class, that picks the classes with positive score; (ii) a set of one-vs-one classifiers, one per every pairwise class combination, that collects the votes or scores from each pairwise classifier and picks the class with the highest score; (iii) the directed acyclic graph support vector machine (DAG-SVM) that applies a set of binary SVMs with over a decision directed acyclic graph. We will discuss how each one works and how each model is trained if we assume $k \in C_k$ classes and datasets $(x_n, y_m)$ where $n >> m$ and $m \leq k$. Figure 2.8 shows data from only three classes for the sake of simplicity.
2.7.1 One-vs-all classifier

Rifkin [34] designed a one-vs-all classifier which constructs $k$ models, $w$, of binary SVM (one model for each class). The training of model $w_k$ uses the training set where the data from class $c_k$ are labelled with $+1$ and the data from the other classes with $-1$. All binary classifiers in a one-vs-all classifier would compute the score of the input data in turn in the inference. In most cases, only one class $y_k(x)$ will have a positive score, and the remaining $k - 1$ will have negative scores. In Figure 2.9, the one-vs-all classifier is split into three binary classifiers to predict Class 1, Class 2 and Class 3 respectively.

The drawback of the one-versus-all approach is that the training datasets are imbalanced.
Chapter 2. Review of Related Works

2.7.2 One-vs-one classifier

The one-vs-one classifier [35] is a technique to train $k(k - 1)/2$ binary classifiers for a multi-class classifier with $k$ classes. It can be written as:

$$w^*, b^* = \arg\min_{w, b} \left( \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{K} \xi_i \right)$$

subject to

$$w^T y_i x_i + b_y - (w^T_k x_i + b_k) \geq 1 - \xi_i, \forall k \neq y_i, \xi_i \geq 0, i = 1...K$$

Each binary classifier is trained against the data of a pair of classes from the training set. During the inference, the test sample will be used as input to the $k(k - 1)/2$ binary classifiers which output "+1" or "-1" respectively. The class obtained with the highest score is selected as the classified label. Figure 2.10 shows how the one-vs-one classifier splits the multi-class problem into a set of binary classifiers.

This approach, too, suffers from ambiguities in that some regions of its input space may receive the same number of votes, see Figure 2.11a. Furthermore, for a large number of
classes $k$, it requires significantly more training time and prediction time than the one-versus-all approach.

### 2.7.3 Directed Acyclic Graph-support vector machines

The Directed Acyclic Graph support vector machines (DAGSVM) was proposed in [36] and is a refinement of the one-vs-one method. The prediction procedure of DAGSVM follows a binary search tree. Each node of the tree is a one-vs-one classifier between classes $c_i$ and $c_j$ and moves to either left or right until a leaf node is reached. Figure 2.11b gives an example of DAGSVM classification with three classes. Let us assume to have a sample $x$ from class $c_1$: the search starts at the root node and the one-vs-one classifier of classes 1 and 3 predicts label “not 3”. Then the samples moves to the left edge and is tested by the one-vs-one classifier of classes 1 and 2 which predicts label “not 2”. Eventually, the leaf node will indicate the predicted class.

The training of the DAGSVM is the same as the one-vs-one method, requiring learning $k(k-1)/2$ binary classifiers. During the prediction, the sample goes through a binary search tree which has $k(k - 1)/2$ internal nodes and $k$ leaves, requiring only $k$ classifications.
The great advantages of using a DAG are that the prediction is faster than with the one-vs-one method and the output label is never ambiguous. It is an elegant method that could be applied to any binary reduction of a multi-class problem.

2.8 The Structural Support Vector Machine

The structural support vector machine (SSVM) was proposed by Tsochantaridis et al [15] as a classification approach that can deal with more complex output objects, such as trees, sequences, sets and more general graphs. To understand this case in intuitive terms, consider the problem of classifying all the objects in an image. This can be approached by taking patches of the image and classifying them one by one separately. However, if one patch is classified as “whiteboard”, it becomes more likely that the object next to it could be a desk, beyond the mere evidence provided by its patch. The separate classification approach would fail to capture this relationship. Conversely, structural (or structured) classification will take into account such mutual dependencies, predicting all the objects in the image jointly and therefore harbouring the potential for higher accuracy. In general, structural classification involves multiple, dependent output variables, structured output spaces, and class attributes. To address these requirements, structural SVM provides a maximum-margin formulation and a cutting plane algorithm which can solve the SVM quadratic program with a potentially prohibitive number of constraints.

The general problem of learning is based on a training sample of input-output pairs $(x_1, y_1), \ldots, (x_n, y_n) \in X \times Y$ from an unknown probability distribution, where $x_i$ here denotes an input vector and $y_i$ is the output vector of all the labels associated with $x_i$. The overall goal of SSVM is to learn a scoring function $F : X \times Y \rightarrow \mathbb{R}$ that can assign a suitable score to every input-output pair. As usual, such a function is fully identified by a set of parameters, $w$, that will be determined during training. Once the model is trained, the predicted label vector, $y$, for a given input, $x$, will be found by maximising $F$. This argmax is sometimes called the SSVM discriminative function and noted as $f$.

Considering an example of natural language parsing ([37]), function $f$ maps a given sentence $x$ to a parse tree $y$ as in Figure 2.12. The classification output $y$ contains a label for each
word in the sentence as one of "Noun (N)" , "Verb (V)" or "Determiner (Det)" , and further
tables for the nodes in the higher levels of the parse tree. The general form of function \( f \) can be defined as:

\[
f(x; w) = \arg\max_{y \in Y} F(x, y; w) = \arg\max_{y \in Y} \langle w, \Psi(x, y) \rangle
\]  

(2.31)

Here \( w \) denotes the model’s parameter vector and \( \Psi(x, y) \) is a feature map that suitably
combines the values of \( x \) and \( y \) so that \( w^T \Psi(x, y) \) can provide the score. This map can be
obtained directly from the graph that models the dependencies between the output vari-
ables. Hereafter, we briefly describe the linguistic example while in the following chapters
we will describe the application to time warping in great detail.

In Figure 2.12, we can choose \( F \) such that we get a model that is a so-called isomorphic
context-free grammar. In this model, each node in a parse tree \( y \) for sentence \( x \) corresponds
to a grammar rule. All valid parse trees \( y \) (i.e. trees with a designated start symbol \( S \) as
the root and the words in sentence \( x \) as the leaves) are scored as the sum of the scores
of their nodes. We thus can write this score into the solution of (2.31) where \( \Psi(x, y) \)
denotes a histogram vector of counts (how often each grammar rule occurs in tree \( y \)). The
discriminative function, \( f \), can be efficiently computed by finding the structure \( y \in Y \) that
maximizes \( F(x, y; w) \) via the CKY (from authors Cocke, Younger and Kasami) algorithm
([37]).

Training of the SSVM is, as usual, based on a training set of input vectors and corresponding
ground-truth label vectors. The goal is to make model \( w \) assign higher score to the correct
output \( y_i \) than any other labelings, possibly with a large margin. Such a condition of zero
training error or arbitrary margin can then be compactly written as a set of \( n \vert Y \vert - n \) linear
constraints by \( \vert Y \vert - 1 \) linear inequalities (2.33)

\[
\forall i \in \{1, ..., n\} : \max_{\substack{y \in Y \setminus y_i}} \{ \langle w, \Psi(x_i, y) \rangle \} < \langle w, \Psi(x_i, y_i) \rangle
\]  

(2.32)
Like for the conventional SVM, the SSVM, too, minimizes a trade-off between a term related to the classification loss over the training set \((x, y)\) and a regularization term over \(w\). The score of the correct label vector, \(y_i\), must “beat” that of the closest rival, \(\bar{y}_{i}(w) = \text{argmax}_{y \neq y_i} \langle w, \Psi(x_i, y_i) \rangle\) by an arbitrary margin (typically, 1) while the model’s norm is kept as low as possible. The SSVM objective can therefore be written as follows:

\[
\min_{w} \frac{1}{2} \|w\|^2
\]
\[
\forall i, \forall y \in Y \setminus y_i : \langle w, \delta \Psi_i(y) \rangle \geq 1
\]
\[
\delta \Psi_i(y) = \Psi(x_i, y_i) - \Psi(x_i, y)
\]

Considering again the case of noise and inseparable data, we hereby introduce slack variables and move to optimize a soft-margin criterion. A penalty term in the objective accrues the margin violations over the entire training set:
Alternatively, the margin violations can be penalised by a quadratic term $\frac{C}{2n} \sum_i \xi_i^2$, giving another soft-margin solution:

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

(2.36)

Like in the conventional SVM, $C > 0$ is a constant that controls the trade-off between training error minimisation and margin maximisation.

Another improvement to the model above was provided by Taskar et al [38] and Tsochantaridis et al [15]. The idea is that, for any input $x_i$, the margin imposed in the constraints should not be the same (i.e., 1) for all its possible predictions $y$. The most “undesirable” predictions should be kept at a larger margin, to build a classifier that is unlikely to produce them at run time. Conversely, nearly-correct predictions could be kept at a lower margin. There is a very simple way to choose such a graded margin: using the evaluation loss, $\Delta(y_i, y)$, a function that quantifies the loss associated with prediction $y$ if ground truth is $y_i$. Typically, $\Delta(y_i, y_i) = 0$ and for $y \neq y_i, \Delta(y_i, y) > 0$. This approach is called margin rescaling. With these changes, the soft-margin solution takes the following form:

$$\begin{align*}
\min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{2n} \sum_{i=1}^n \xi_i^2 \\
s.t. \forall i, \forall y \in Y \setminus y_i : \langle w, \delta \Psi_i(y) \rangle \geq \Delta(y_i, y) - \xi_i, \xi_i \geq 0
\end{align*}$$

(2.37)
In (2.37), the constraints ensure that the score assigned to the ground-truth label vector, $y_i$, is higher than that assigned to any other label vector, $y$, by a margin equal to the classification loss for that prediction, $\Delta(y_i, y)$. At its turn, $\Delta(y_i, y)$ is a loss function that can be arbitrarily chosen to quantify the inaccuracy of incorrect predictions.

The challenge with SSVM is that the number of possible alignments for a given sequence pair is exponential in their length. This in turn leads to a highly-constrained learning objective that proves computationally infeasible even for relatively short sequences. However, Tsochantaridis et al. in [15] have shown that a very close, controlled approximation to the solution of (2.37) can be obtained by using only a polynomial (i.e., easily feasible) number of constraints. The constraints are chosen as the “most-violated constraints”, i.e., the constraints that set the value of variable $\xi_i$ for each sample, $i = 1 \ldots N$. Assuming that $\Delta(y_i, y) > 0$ for $y_i \neq y$ and $\Delta(y, y) = 0$, Let us re-consider the constraints in (2.37) and re-arrange their terms:

\[
\begin{align*}
  w^T \psi(x_i, y_i) - w^T \psi(x_i, y) &\geq \Delta(y_i, y) - \xi_i \quad \forall y \\
  \Rightarrow \xi_i &\geq -w^T \psi(x_i, y_i) + w^T \psi(x_i, y) + \Delta(y_i, y) \quad \forall y \\
  \Rightarrow \xi_i &\geq \max_y (-w^T \psi(x_i, y_i) + w^T \psi(x_i, y) + \Delta(y_i, y)) \\
  \Rightarrow y_i &\geq \arg \max_y (w^T \psi(x_i, y) + \Delta(y_i, y)) \\
\end{align*}
\]  

With the changes above, it is very common to see the objective of margin-rescaling structural SVM rewritten in the compact form: $\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max_{y \in Y} \{ \Delta(y_i, y') + w \cdot \Phi(x_i, y') - w \cdot \Phi(x_i, y_i) \}$. Fortunately, this problem enjoys a number of robust and efficient solvers that can be used off-the-shelf such as Joachims’ SVMstruct (used extensively in this thesis) and PyStruct.
Chapter 3

Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos

3.1 Introduction

When dealing with sequential data, one of the urgent problems is how to align multiple sequences to allow their meaningful comparison. This problem, known as sequence alignment or warping, concerns fields as diverse as bioinformatics, finance, climate series analysis and meteorology, and multimedia signal processing at large. The problem is often framed as the alignment of two given sequences, with the first being used as reference and the second being aligned, or “warped”, onto the first. In the case of video clips, the goal of sequence alignment is that of finding corresponding frames in two given videos to be used for comparison, analysis and, possibly, classification.

The most well-known sequence alignment technique is dynamic time warping (DTW). Its main idea is to scan both sequences while looking for local correspondences of minimum cost, where the cost is a function that reflects the similarity between both the frames and their indices [1]. The outputs of DTW are a path, i.e., a set of index correspondences in the two sequences, and a total cost which can be interpreted as an overall dissimilarity
between the sequences. DTW is an instance of dynamic programming algorithms and, as such, the returned path is guaranteed to be globally optimal. While DTW was originally proposed for the alignment of time series, it has later found use in a number of other applications including data mining [39], speech processing [7], medicine [40] and classification of genomic signals [8]. Over the years, many extensions have been proposed. Windowing DTW restricts the corresponding frames to fall within a given window [9]. Slope-weight DTW restricts the search to paths within a given slope [39]. Keogh and Pazzani [11] used derivatives of the original signal to improve the alignment before applying DTW. In computer vision, Gong and Medioni have extended DTW by integrating it with manifold learning [41]. Hsu et al. have augmented warping along the time dimension with smooth spatial warping to align actions performed with different styles [42]. Gritai et al. have exploited anthropometric and epipolar constraints to improve the alignment of human actions [43]. Junejo et al. have used DTW to recognize human actions under view changes. Amongst the many algorithms, the state of the art is likely held by the generalized canonical time warping (GCTW) that applies canonical correlation analysis (CCA) alongside DTW to perform the alignment in a subspace [12]. The role of CCA is to analyze the two sets of multivariate measurements and extract the most informative linear combinations of their dimensions [44]. GCTW iteratively alternates between CCA and a weighted version of DTW to simultaneously find an optimal linear subspace and a path, and it has recently reported the best performance over a variety of video datasets [12].

However, DTW and GCTW likewise are limited to linear cost models and do not provide explicit procedures for the training of the model’s parameters. In other words: can we learn how to automatically align two sequences from a training set of manually-aligned sequence pairs? This view of the alignment problem places it more clearly in the realm of machine learning. An improved model for the alignment of sequences is then offered by the pair hidden Markov model (pair HMM, or PHMM for short) that is, at the same time, a variant of DTW and of a conventional HMM providing a full probabilistic treatment of the alignment problem [14]. PHMM is a generative sequential model that emits pairs of measurements and allows for model training under maximum likelihood or other estimation frameworks.
In this chapter, we propose a novel alignment algorithm that combines the features of a PHMM with the practical parameter estimation of the structural support vector machine (SSVM) [15]. The new model - named pair hidden Markov support vector machine (PHMM-SSVM) - offers several advantages over conventional alignment algorithms including: 1) the ability to learn the cost model from any sets of supervised examples; 2) the use of a maximum-margin training objective that has a proven reputation for accurate prediction; (3) the flexibility of using any loss functions of choice during training. Our main contributions are a set of dedicated feature and loss functions that allow PHMM-SSVM to achieve remarkable alignment accuracy. The proposed model has been tested against DTW and GCTW in a set of experiments on action alignment over pairwise versions of the Weizmann dataset [45] and the Olympic Sports dataset [46]. The experimental results show that the proposed approach can outperform existing models in terms of alignment accuracy.

3.2 The Proposed Model: PHMM-SSVM

3.2.1 Pair Hidden Markov Model

PHMM is a probabilistic model for pairwise sequence alignments. Given two sequences, \( s = \{s_1, ..., s_i, ..., s_L_s\} \) and \( t = \{t_1, ..., t_j, ..., t_L_t\} \), their alignment can be intuitively defined as a sequence of index pairs from the two sequences. However, to simplify both notations and operations, the alignment is re-defined as a sequence of only three types of symbols: \( M \) (“match”), \( S \) (“insert a gap on sequence \( s \)”) and \( T \) (“insert a gap on sequence \( t \)”). The symbols have the following meaning: assuming \( i \) and \( j \) to be the current indices over sequences \( s \) and \( t \), respectively, 1) symbol \( M \) pairs frames \( s_i \) and \( t_j \) and then increments both indices; 2) symbol \( S \) pairs no frames and only increments index \( j \); and, likewise, 3) symbol \( T \) pairs no frames and only increments index \( i \). As a toy example, we show below a possible alignment for two short sequences from character set \( \{A, B, C, D\} \):
Chapter 3. Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos

\[ s = A \quad B \quad C \quad B \quad D \quad A \quad D \]
\[ t = A \quad B \quad D \quad C \quad B \quad A \quad D \]
\[ y = M \quad M \quad S \quad M \quad M \quad T \quad M \quad M \]  (3.1)

In the above example, sequence \( y \) encodes the alignment, with the \( M \) symbols showing the matched frames (e.g., \( s_3 \) and \( t_4 \)) and the \( S \) and \( T \) symbols accounting for the required gaps. The length of the alignment is bounded between \( \max(L_s, L_t) \) and \( L_s + L_t \).

In probability notation, a PHMM is a model for the joint probability, \( p(s, t, y) \), of the two sequences and their alignment. Such a model can be used to infer an optimal alignment, \( \tilde{y} \), for the two sequences as \( \tilde{y} = \arg\max_y p(s, t, y) \). Like for a conventional HMM, the joint probability of a PHMM factorises into a set of transition and emission probabilities. The transition probabilities are commonly defined as: (1) \( \delta \) for transitions from \( M \) to either \( S \) or \( T \); (2) \( \varepsilon \) for staying in \( S \) or \( T \); (3) \( 1 - \varepsilon \) for transitions from either \( S \) or \( T \) to \( M \). Note that the model bars direct transitions from \( S \) to \( T \) and vice versa assuming that a pair of matched frames will always follow a run of gaps. Figure 3.1 shows the state diagram of the PHMM, while Table 3.1 shows the complete transition probabilities table.

We note the probability of emitting aligned pair \((a, b)\) as \( p_{a, b} \) and the probability of emitting measurement \( a \) against a gap as \( q_a \). In the common case of numerical measurements, both \( p \) and \( q \) will be multi-variate likelihoods such as Gaussian distributions or mixture models.

Using a PHMM, the optimal alignment for a pair of sequences can be found via an equivalent Viterbi algorithm [25]. Its computational complexity, \( O(L_s L_t) \), is only linear in the length of each sequence thus ensuring fast and efficient alignments. The main steps of the algorithm are given below, where the probability of reaching state \( * = \{M, S, T\} \) at indices \( i \) and \( j \) over \( s \) and \( t \) is noted as \( p^*(i, j) \).

**Initialization**: \( p^M(1, 1) = p^S(1, 1) = p^T(1, 1) = p^*(0, j) = p^*(i, 0) = 1 \).

**Recurrence**: \( i = 1, ..., L_s, j = 1, ..., L_t \):
Chapter 3. *Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos*  

\[ p^M(i, j) = \max_{p_{s,t,j}} \begin{cases} 
(1 - 2\delta) p^M(i - 1, j - 1) \\
(1 - \varepsilon) p^S(i - 1, j - 1) \\
(1 - \varepsilon) p^T(i - 1, j - 1) 
\end{cases} \tag{3.2} \]

\[ p^S(i, j) = \max_{q_{s,i}} \begin{cases} 
\delta p^M(i - 1, j) \\
\varepsilon p^S(i - 1, j) 
\end{cases} \tag{3.3} \]

\[ p^T(i, j) = \max_{q_{t,j}} \begin{cases} 
\delta p^M(i, j - 1) \\
\varepsilon p^T(i, j - 1) 
\end{cases} \tag{3.4} \]

Termination:

\[ p(s, t, y) = \max(p^M(L_s, L_t), p^S(L_s, L_t), p^T(L_s, L_t)) \tag{3.5} \]

### 3.2.2 Structural SVM

Structural SVM is a powerful classifier that extends the notion of maximum-margin classification to the case of structured prediction. This case includes the classification of structures such as sequences and graphs, and problems such as alignment and ranking. In the case of alignment, the problem is to learn a scoring function, \( F(s, t, y) \), between input sequences \( s \) and \( t \) and output alignment \( y \) based on training samples of input-output pairs. The scoring function typically takes the form of a linear discriminant, \( F(s, t, y) = w^T \psi(s, t, y) \), that can be extended to non-linear mappings by the use of kernels. To learn an accurate alignment predictor, the training objective ensures that, for every training sample, the scoring function assigns its ground-truth alignment with a score higher than that of any other alignments by an appropriate margin.

The challenge with structural SVM is that the number of possible alignments for a given sequence pair is exponential in their length. This in turn leads to a highly-constrained learning objective that proves computationally infeasible even for relatively short sequences.

However, Tsochantaridis *et al.* in [15] have shown that a very close approximation to the solution of SSVM can be obtained by using only a polynomial (i.e., easily feasible) number of constraints, and Joachims *et al.* in [47] have shown that this approach can also be used
for the sequence alignment problem. Given a supervised training set of sequence pairs and alignments, \((s^i, t^i, y^i)\), \(i = 1 \ldots N\), the relaxed objective can be written as:

\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi^i \quad \text{s.t.} \\
w^\top \psi(s^i, t^i, y^i) - w^\top \psi(s^j, t^j, y) \geq \Delta(y^i, y) - \xi^i, \\
i = 1 \ldots N, \ \forall y \in \mathcal{W}
\] (3.6)

Like for a conventional SVM, objective (3.10) aims to strike a balance between the prediction error over the training set (\(\sum_{i=1}^{N} \xi^i\)) and a regulariser (\(\|w\|^2\)). The constraints ensure that the score assigned to the ground-truth alignment, \(y^i\), is higher than that assigned to any other alignment \(y\) stored in a working set, \(\mathcal{W}\), by margin \(\Delta(y^i, y)\). At its turn, \(\Delta(y^i, y)\) is a loss function that can be arbitrarily chosen to quantify the inaccuracy of incorrect alignments. The working set, \(\mathcal{W}\), is populated using a constraint-violation approach that ensures that the solution for the relaxed objective (3.10) is “epsilon-close” to the solution of the complete objective. More details are provided in Section 3.2.6.

### 3.2.3 Integration

The integration of the PHMM in the SSVM framework (PHMM-SSWM hereafter) can be obtained by simply setting the PHMM’s joint probability as:

\[
p(s, t, y) \propto \exp(w^\top \psi(s, t, y))
\] (3.7)

This restricts the emission probabilities to belong to the exponential family of distributions, which is however a very broad and encompassing family (Gaussian, Gamma, chi-squared etc). In addition, the assumption does not require the distribution to be normalised and therefore the \(w\) parameters can be chosen from a larger domain. Lastly, a non-linear parametrisation can be easily obtained by using kernels.
3.2.4 Parameter Vector and Feature Function

In the structural SVM framework, the score for a sample \((s, t, y)\) is obtained from the product of a parameter vector, \(w\), and a feature function, \(\psi\), that provides a re-mapping of the given measurements and labels. As a common assumption, the score is assumed to be a decomposable function (a sum) over the individual labels of the assignment, \(y_k, k = 1 \ldots |y|\).

The parameter vector contains two sections: transition parameters \(w^{tr}\) and emission parameters \(w^{em}\). The transition states include "MATCH (M) BETWEEN \(s_i\) AND \(t_j\)", "INSERT A GAP (\(I_s\)) ON \(S\)", "INSERT A GAP (\(I_t\)) ON \(T\)". Figure 3.1 shows the state diagram of the PHMM, while Table 3.1 shows the complete transitions table. The transition parameters \(\delta, \tau, \varepsilon\) lead a \(4 \times 4\) transition matrix indexed by labels \(y_{k-1}\) and \(y_k\). Transitions between symbols \(I_s \rightarrow I_t\) and \(I_t \rightarrow I_s\) are blocked because the operation "Insert a gap" only can be happened on the sequence itself.

As emission features, we simply consider the absolute difference of measurements \(s_i\) and \(t_j\) in matching states; therefore, the emission parameters are a vector with the same dimensionality as the individual measurements, i.e. \(w^{em}, s_i, t_j \in \mathbb{R}^D\). Logically, \(w^{em}\) should be assigned negative values during training so that more dissimilar measurements receive lower scores; however, we do not impose a negativity constraint on these parameters. With these assumptions, the score can be re-written as:

\[
\text{Score} = \sum_{k} \left( w^{tr}_{y_{k-1}, y_k} \psi(s_{k-1}, t_k) + w^{em}_{s_{k-1}, t_k} \right)
\]
\[ w^\top \psi(s, t, y) = \sum_{k=1}^{\vert y \vert} w^t_{y_{k-1}, y_k} + w^e_{m, n} \vert s_i - t_j \vert I[y_k = M] \]

\[ w^t_{0, s} = 0; \quad I[y_k = M] : i++, j++; \]

\[ I[y_k = S] : j++; \quad I[y_k = T] : i++ \]

where \( I \) is the indicator function and indices \( i \) and \( j \), initially set to 1, are post-incremented according to the value of label \( y_k \).

### 3.2.5 Loss Functions

The common way to measure the inaccuracy of a predicted alignment is by use of a Hamming distance between the ground-truth alignment, \( y \), and the prediction, \( \bar{y} \). This function is often referred to as \( Q \)-loss function in the alignment literature and noted as \( \Delta_Q(y, \bar{y}) \) [48]. The \( Q \)-loss is decomposable over the individual operations in the alignments as \( \Delta_Q(y, \bar{y}) = 1 - \sum_1^{\vert y \vert} \delta(y_k, \bar{y}_k) \). At its turn, \( \delta(y_k, \bar{y}_k) \) returns \( 1/N \) (\( N \): number of frame matches in the ground truth) when a ground-truth match is correctly predicted and 0 otherwise. In practice, we compute the loss by explicitly unfolding all the frame indices over sequences \( s \) and \( t \) in both the ground truth and the predicted alignment.

Another useful loss function is the \( Q_4 \)-loss: this is a more lenient loss function that counts a match as correct even if the indices of the matching frames in the prediction are shifted by \( \pm 2 \) compared to those in the ground truth. In the experiments, we report results in terms of both \( Q \)-loss and \( Q_4 \)-loss. In addition, during the annotation of the training set we annotate the ground-truth alignment only for some “key” frames that we can match with high confidence (e.g., apex phases of movements). The loss is measured only against such key-frame matches.

### 3.2.6 Most-Violated Constraints

The learning problem in (3.10) requires that, for each training sample, the ground-truth alignment is given a score higher than any other alignments. Empirical minimum risk
(EMR) classifiers learn the classifier’s parameters by minimizing a chosen loss function over a given training set. To avoid over-fitting the model onto the training data, regularization terms are also often added to the minimization objective. The most famous member of ERM is the support vector machine which has also been extended to the case of structured prediction, i.e., the classification of structures such as sequences and graphs, and tasks such as ranking and alignment (structural SVM [15]). In the alignment case, the problem is to learn a scoring function, $F(s, t, y)$, that quantifies the compatibility of measurement sequences $s$ and $t$ and alignment path $y$ based on training samples of manually-aligned sequence pairs. $\psi$ is a function that maps an alignment $y$ of $s$ and $t$ to a so-called feature vector (a suitable numerical vector that is independent of the model’s parameters; details are provided in the following subsection), and $w$ is the vector of the model’s parameters. As the equation 3.7 shows, PHMM can be represent the exponential family as a linear model that is completely equivalent to the full probabilistic model.

Given a model, $w$, inference of the optimal alignment path, $\bar{y}$, for two given input sequences can thus be formally obtained as:

$$\bar{y} = \arg\max_{y \in Y} [w^T \psi(s, t, y)] \quad (3.9)$$

where $Y$ is the set of all possible alignments between $s$ and $t$. If the score function, $w^T \psi(s, t, y)$, can be decomposed as a sum over the frames of the sequences, its maximum in $y$ can be efficiently found using a dynamic programming algorithm equivalent to the modified Viterbi algorithm (3.2) in logarithmic scale.

At its turn, the model can be learned by minimising the risk over a given training set of supervised sequences-paths using structural SVM. By noting the training set as $(s^i, t^i, y^i), i = 1 \ldots N$, the learning objective of structural SVM can be written as:
Chapter 3. Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos

\[
\arg\min_{w,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi^i \quad \text{s.t.} \\
w^\top \psi(s^i, t^i, y^i) - w^\top \psi(s^i, t^i, y) \geq \Delta(y^i, y) - \xi^i; \\
i = 1 \ldots N, \forall y \in \mathcal{Y} 
\]  

(3.10)

Like for a conventional SVM, objective (3.10) aims to minimise a trade-off between a term related to the classification loss over the training set (\(\sum_{i=1}^{N} \xi^i\)) and a regularization term (\(\|w\|^2\)). The constraints ensure that the score assigned to the ground-truth alignment, \(y_i\), is higher than that assigned to any other alignment, \(y\), by a margin equal to the classification loss for that alignment, \(\Delta(y^i, y)\). At its turn, \(\Delta(y^i, y)\) is a loss function that can be arbitrarily chosen to quantify the inaccuracy of incorrect alignments.

The challenge with structural SVM is that the number of possible alignments for a given sequence pair is exponential in their length. This in turn leads to a highly-constrained learning objective that proves computationally infeasible even for relatively short sequences. However, Tsochantaridis et al. in [15] have shown that a very close, controlled approximation to the solution of (3.10) can be obtained by using only a polynomial (i.e., easily feasible) number of constraints, and Joachims et al. in [47] have shown that this approach can also be used for the sequence alignment problem. The constraints are chosen as the “most-violated constraints”, i.e., the constraints that set the value of variable \(\xi^i\) for each sample, \(i = 1 \ldots N\). Let us consider the constraints in (3.10) and rearrange their terms:

\[
w^\top \psi(s^i, t^i, y^i) - w^\top \psi(s^i, t^i, y) \geq \Delta(y^i, y) - \xi^i \quad \forall y \\
\rightarrow \xi^i \geq -w^\top \psi(s^i, t^i, y^i) + w^\top \psi(s^i, t^i, y) + \Delta(y^i, y) \quad \forall y \\
\rightarrow \xi^i = \max_y (-w^\top \psi(s^i, t^i, y^i) + w^\top \psi(s^i, t^i, y) + \Delta(y^i, y)) \\
\rightarrow y^{\xi^i} = \arg\max_y (w^\top \psi(s^i, t^i, y) + \Delta(y^i, y)) 
\]  

(3.11)

Equation (3.11) shows that the alignment \(y^{\xi^i}\) setting the value of variable \(\xi^i\) can be found by a modified version of the inference, known as the “loss-augmented” inference since it
adds up the loss function to the score. If the loss function, too, can be evaluated frame-by-frame, the maximum of the loss-augmented inference can still be found efficiently by a suitably-weighted Viterbi algorithm.

Algorithm 3 shows the main steps of the training procedure. In the pseudo-code, $\epsilon$ is a small constant that sets the accuracy of the approximation (set to 0.01 in the experiments), and $W$ is the set of the most-violated constraints. As for similar quadratic programs, the training algorithm enjoys convergence to a global optimum [15].

**Algorithm 3:** Structural SVM training algorithm: main steps.

**Input:** Measurement sequences $s^i, t^i$ and ground-truth alignment $y^i$, $i = 1 \ldots N$; parameter $\epsilon$

$W = \emptyset$, $w = 0$, $\xi = 0$

repeat

foreach $i = 1 \ldots N$ do

$y^{*i} \leftarrow \operatorname{argmax}_y (w^T \psi(s, t, y) + \Delta(y^i, y));$

if $\xi^i = \left[ w^T (\psi(s^i, t^i, y^{*i}) - \psi(s^i, t^i, y^i)) + \Delta(y^i, y^{*i}) \right] > \xi^{i \text{prev}} + \epsilon$ then

$W \leftarrow W \cup y^{*i};$

end

end

$(w, \xi) = \operatorname{argmin}_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi^i \ s.t. \ W;$

until $\xi$ unchanged;

**Output:** Model $w$

### 3.3 Experiments

The following experiments evaluate the proposed PHMM-SSVM model in the temporal alignment of action videos against DTW [1] and a state-of-the-art algorithm, GCTW [13]. In the first experiment, we compare the performance in aligning the “jump” action (fig 3.2a) from different subjects of the Weizmann dataset [45]. In the second experiment, we compare the “clean-and-jerk” action (fig 3.2b) performed by 11 subjects from the challenging Olympic Sports dataset [46]. For the SSVM training, we have set parameters $C$ to 10 and $\epsilon$ to 0.01, with no noticeable sensitivity. Results are reported in terms of both $Q$-loss and $Q_4$-loss (see section 3.2.5).
Chapter 3. Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos

3.3.1 Results on the Weizmann Dataset

The Weizmann dataset contains ten actions performed by nine actors. While it has been long saturated in terms of action recognition accuracy, according to [13] it is still probing for testing alignment accuracy. In this experiment, we follow [13] and first subtract the background from the videos and then process the resulting frames by the Euclidean distance transform [49], preserving 99% of the energy by retaining the top 416 principal components.

As test data, we have formed 13 video pairs from action “jump” selecting different subject pairs and annotating the ground-truth alignments manually. We have then randomly picked 6 as training set and the others as test set, yet ensuring that the subjects pairs in the test set did not appear in the training.

Table 3.2 reports the alignment accuracy on the test set as the one-complement of the $Q$ and $Q_4$ losses. The table clearly shows that PHMM-SSVM achieves higher accuracy in terms of $Q$-loss than both DTW (27.4 percentage points) and GCTW (7.8 percentage points). Since the $Q_4$-loss is more lenient, its accuracy is generally higher for all algorithms; however, the proposed PHMM-SSVM still achieves the highest accuracy and the ranking is unvaried.

Table 3.2: Alignment accuracy for action “jump” in the Weizmann dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$1 - Q$-loss (%)</th>
<th>$1 - Q_4$-loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHMM-SSVM</td>
<td>68.6%</td>
<td>98.0%</td>
</tr>
<tr>
<td>DTW</td>
<td>41.2%</td>
<td>72.6%</td>
</tr>
<tr>
<td>GCTW</td>
<td>60.8%</td>
<td>96.1%</td>
</tr>
</tbody>
</table>

Figure 3.2: Sample actions
3.3.2 Results on the Olympic Sports Dataset

The Olympic Sports dataset is a more challenging dataset of real sport videos from YouTube. In this dataset, we chose action “clean-and-jerk” (a specialty of weightlifting) since the manual alignment is relatively certain. We created 55 pairs of ground-truth alignments and split them into 27 pairs for training and 28 for test. As measurements, we computed dense feature descriptors for each frame of the video sequences using Laptev’s STIP extractor [50]. We then computed a bag-of-words for each frame with 1,000 bins using the VLFeat library [51]. Note that the specific choice of features is not the focus of this paper.

Table 3.3 reports the alignment accuracy on the test set. Again, the table clearly shows that PHMM-SSVM achieves higher accuracy in terms of $Q$-loss than both DTW (13.1 percentage points) and GCTW (7.6 percentage points). While the $Q_4$-loss tends to reduce...
Chapter 3. *Pair Hidden Markov Support Vector Machine and its Application for the Alignment of Videos*

<table>
<thead>
<tr>
<th></th>
<th>1 - $Q$-loss (%)</th>
<th>1 - $Q_4$-loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHMM-SSVM</td>
<td>50.0%</td>
<td>74.2%</td>
</tr>
<tr>
<td>DTW</td>
<td>36.9%</td>
<td>69.6%</td>
</tr>
<tr>
<td>GCTW</td>
<td>42.4%</td>
<td>70.5%</td>
</tr>
</tbody>
</table>

Table 3.3: Alignment accuracy for action “clean-and-jerk” in the Olympic Sports dataset.

these differences, PHMM-SSVM still achieves the highest accuracy in terms of $Q_4$-loss and the ranking is again unvaried. Figure 3.3 shows an example of the ground-truth and predicted alignments: a) the top two rows show six manually-matched key-frames from the two sequences. The frames from the first sequence are used as “template” and those from the second represent the ground-truth alignment; b) the third row shows the alignment predicted by the proposed PHMM-SSVM; and c) the bottom two rows show the alignments predicted by GCTW and DTW, respectively. The superimposed ellipses and rectangles visually highlight the alignment errors. This figure shows that the alignment predicted by PHMM-SSVM only mildly differs from the ground truth and is more accurate than those returned by DTW and GCTW.

3.4 Conclusion

In this chapter, we have presented a novel approach for sequence alignment and showed its effectiveness in aligning human actions in videos. The proposed method - named pair hidden Markov support vector machine (PHMM-SSVM) - integrates the probabilistic formulation of the pair HMM with the effective parameter training of structural SVM. The proposed integration includes dedicated feature and loss functions suitable to achieve accurate alignments. Experimental results over two probing video datasets show that PHMM-SSVM achieves higher accuracy than both a standard dynamic programming solution (DTW) and a state-of-the-art algorithm (GCTW), with improvements over the runner-up of more than 7 percentage points. In addition, while in this thesis we have used the proposed model for aligning actions in videos, there are no standing limitations to its general use in any other domain. In the next chapter, we extend our approach to generic data characteristics and to semi-supervised settings with limited ground-truth annotations.
Chapter 4

The Extended Hidden Markov Support Vector Machine and its Application to the Alignment of Human Actions

4.1 Introduction

In chapter 3, we have proposed a PHMM-SSVM alignment framework for video sequences which integrates the probabilistic model of a pair hidden Markov model with the efficient parameter learning of the structural SVM. This new framework provides a more and accurate robust way to align two video sequences. It can outperform a state-of-art alignment algorithm, GCTW, if the model can be trained on an adequate training set of manually-aligned video sequences. However, a PHMM-SSVM model also has limitations: 1) annotating a complete manual ground-truth for a sequence pair can prove time-consuming and ambiguous; 2) the (dis)similarity function used to measure the difference between two matched frames is Euclidean, but other functions may prove more suitable; 3) the crucial step of the loss-augmented inference needs to be modular to a variety of alignment loss functions such as the Q-loss and Q4-loss.
In this chapter, we propose an extension of the PHMM-SSVM framework called the *extended hidden Markov support vector machine* (EHMM-SSVM) which addresses all of the above shortcomings and outperform the state of the art on human action sequences. It includes these original contributions:

- Two dedicated cost functions (one stricter, one more lenient) that can be used to describe desirable performance for the sequence alignment task;
- A performance analysis of various dissimilarity functions for the comparison of video frames;
- Partial ground-truth alignments based on only “key” frames that are suitably interpolated to produce full ground-truth alignments;
- An extensive experimental evaluation that includes actions from two probing datasets, UCF101 and MSR Daily Activity 3D, with both quantitative and qualitative comparisons.

The proposed model has been tested against DTW (as baseline) and GCTW (state of the art) in a set of experiments on action alignment over selected actions from the UCF101 dataset [52] and the MSR Daily Activity 3D dataset [53]. The experimental results show that the proposed approach has been able to outperform the compared approaches by at least 10 percentage points of alignment accuracy in all the experiments.

### 4.2 Extended Hidden Markov Model

The extended HMM for sequence alignment (EHMM) is a model for the joint probability, \( p(s, t, y) \), of the two sequences and their alignment path. It includes all the advantages of a PHMM to infer an optimal alignment between two given sequences, plus additional elements of flexibility. The joint probability of an EHMM factorizes into a set of transition and emission probabilities. The transition probabilities include: (1) the probabilities to transition from state \( M \) to either \( S \) or \( T \); (2) the probabilities to transition from either \( S \) or \( T \) to \( M \); and (3) the probabilities to stay in \( S \) or \( T \). We note these transition probabilities
Table 4.1: Transition probabilities table.

<table>
<thead>
<tr>
<th>Begin</th>
<th>M</th>
<th>S</th>
<th>T</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w_1$</td>
<td>$w_2$</td>
<td>$w_2$</td>
<td>$w_3$</td>
</tr>
<tr>
<td>$M$</td>
<td>$w_4$</td>
<td>$w_5$</td>
<td>$w_5$</td>
<td>$w_3$</td>
</tr>
<tr>
<td>$S$</td>
<td>$w_6$</td>
<td>$w_7$</td>
<td>N/A</td>
<td>$w_3$</td>
</tr>
<tr>
<td>$T$</td>
<td>$w_6$</td>
<td>N/A</td>
<td>$w_7$</td>
<td>$w_3$</td>
</tr>
</tbody>
</table>

Figure 4.1: The extended hidden Markov model for sequence alignment (represented as an undirected graphical model).

Figure 4.2: Main steps of the proposed approach: a) training (with regularized empirical risk minimization, i.e., structural SVM - Section 3.2.6); inference (Section 4.2).

as $p(y_k|y_{k-1})$. EHMM model blocks transitions between the insertion states $I_s$ and $I_t$, thereby prohibiting from inserting unaligned gaps to both sequences. This model also releases the probability constraints on the transitions which allow SSVM to learn the 7 transitions in $w$ parameters in the table 4.1.

To complete the model, we also need to define the emission probabilities. To this aim, we note the probability of emitting a matched pair of measurements, $(a,b)$, as $p_{a,b}$ and the probability of emitting measurement $a$ against a gap as $q_a$. In the common case of numerical measurements, both $p$ and $q$ will be multivariate likelihoods such as Gaussian
distributions or mixture models. Figure 4.1 shows a graphical model representation of the EHMM.

Using an EHMM, the optimal alignment for a pair of sequences can be found via an equivalent Viterbi algorithm [25]. Its computational complexity, \( O(L_s L_t) \), is only linear in the length of each sequence (and quadratic overall), thus ensuring fast alignments up to reasonably long sequences. The main steps of the algorithm are given below, where the probability of reaching state \(* = \{M, S, T\}\) at indices \(i, j\) over \(s\) and \(t\) is noted compactly as \(p^*(i, j)\).

**Initialization:**

\[
\begin{align*}
p^M(1, 1) &= p^S(1, 1) = p^T(1, 1) = p^*(0, j) = p^*(i, 0) = 1.
\end{align*}
\]

**Recurrence:**

\[
\begin{align*}
p^M(i, j) &= p_{s_i, t_j} \max \left\{ \begin{array}{l} w_1 p^M(i - 1, j - 1) \\
                     w_6 p^S(i - 1, j - 1) \\
                     w_6 p^T(i - 1, j - 1) \end{array} \right. \\
p^S(i, j) &= q_{s_i} \max \left\{ \begin{array}{l} w_5 p^M(i - 1, j) \\
                     w_7 p^S(i - 1, j) \end{array} \right. \\
p^T(i, j) &= q_{t_j} \max \left\{ \begin{array}{l} w_5 p^M(i, j - 1) \\
                     w_7 p^T(i, j - 1) \end{array} \right.
\end{align*}
\]

**Termination:**

\[
p(s, t, y) = \max(p^M(L_s, L_t), p^S(L_s, L_t), p^T(L_s, L_t))
\]

4.3 The Proposed Model

The model proposed for the alignment of video pairs consists of: a) a linear score function that embeds the graphical structure of the EHMM; b) various dissimilarity functions that measure the dissimilarity of any two video frames; and c) two loss functions capable of
Chapter 4. Extended Hidden Markov Support Vector Machine and its Application for the Alignment of Human Actions

properly quantifying alignment/misalignment. These components are presented in the following subsections.

### 4.3.1 Scoring Function

In the structural SVM framework, the score for a sample \((s, t, y)\) is obtained from the product of a parameter vector, \(w\), and a feature function, \(\psi(s, t, y)\), that provides a re-mapping of the given sequences and path. Both the parameter vector and the feature function contain two sections: one accounting for the transitions between the states of the alignment path, and one accounting for the emission of the measurements from the two sequences. The transition parameters, noted as \(w^{tr}\), are a \(4 \times 4\) matrix indexed by states \(y_{k-1}\) and \(y_k\) (note that transitions between symbols \(S \rightarrow T\) and \(T \rightarrow S\) are not allowed). The transition features are therefore just indicator functions that take a value of one for states \(y_{k-1}\) and \(y_k\) and zero otherwise. As emission features, we use a dissimilarity function over measurements \(s_i\) and \(t_j\) emitted by a matching state. Therefore, the emission parameters, \(w^{em}\), are a vector with the same dimensionality as the dissimilarity function.

With these assumptions, the score function can be written as:

\[
w^\top \psi(s, t, y) = \sum_{k=1}^{\vert y \vert} w^{tr}_{y_{k-1}, y_k} + w^{em} d(s_i, t_j) I[y_k = M]
\]

\[
w^{tr}_{0, *} = 0; \quad I[y_k = M] : i++, j++; \\
I[y_k = S] : j++; \quad I[y_k = T] : i++
\]

The notations in (4.5) read as: \(|y|\) is the length of the alignment path; \(d(s, t_j)\) is the dissimilarity function between measurements \(s_i\) and \(t_j\) (details in Section 4.3.2); \(I\) is an indicator function that takes a value of one when its argument is true; and indices \(i\) and \(j\), initially set to 1, are post-incremented according to the value of state \(y_k\). It is evident that the score function decomposes over the individual states of the alignment path, \(y_k, k = 1 \ldots \vert y \vert\), in a form of the type \(\sum_{k=1}^{\vert y \vert} w^\top \phi(s, t, y_k)\), thus permitting a Viterbi-style inference.
4.3.2 Dissimilarity Functions

To quantify the dissimilarity of two frames, we have employed three popular dissimilarity measurements: the cosine distance, the Euclidean distance and the Euclidean squared distance. The cosine distance is defined as one minus the cosine between two vectors [54]. As such, it ranges between zero (the vectors have the same orientation) and two (the vectors have opposite orientation, i.e., are most dissimilar). The Euclidean distance and the Euclidean squared distance measure the dissimilarity as the element-wise sum of squared differences between the two vectors, with and without a final square root, respectively. These distances are increasingly sensitive to the difference in magnitude between the two vectors, and the choice between them should reflect whether this difference is informative or not. In particular, the cosine distance is completely insensitive to the variations in the magnitude of the two input vectors and can therefore focus on differences in direction.

In the experiments described in Section 4.4, we have used two different datasets. For the second dataset, we have utilized specific measurements (3D skeletons) and the differences in magnitude seemed a-priori important. In fact, the Euclidean distance delivered the highest accuracy. For the first dataset, we have used bag-of-words histograms of dense HOG/HOF features [55]. Since these histograms are normalized to a unit magnitude, the cosine distance seemed the most appropriate, and the experimental results have confirmed it.

4.3.3 Loss Function and Loss-Augmented Inference

The loss function, $\Delta(y^a, y)$, assigns a penalty for predicting alignment $y$ when the annotated ground-truth alignment is $y^a$. This function must be able to gradually quantify what we regard as a “bad” or a “good” prediction. In turn, the reciprocal of the loss function can be used as the main measurement of accuracy. In our model, we have adopted two types of loss functions, nicknamed as $Q$-loss and $Q_4$-loss.

To describe these loss functions, we first need to expand each match state of an assignment into its corresponding index pair. We then introduce an indicator function,
Chapter 4. Extended Hidden Markov Support Vector Machine and its Application for the Alignment of Human Actions

$\mathbf{1}[y_h^o = (m, n), y_k = (i, j)]$, that takes value one if the index pairs in its arguments are the same, and zero otherwise. We use this function to define a partial matching function:

$$\delta(y^o, y_k) = \sum_{h=1}^{\lvert y^o \rvert} \mathbf{1}[y_h^o = (m, n), y_k = (i, j)]$$

(4.6)

that checks whether index pair $(i, j)$ in the predicted assignment matches any index pair in the ground-truth assignment. Since indices are allowed to only appear once in an assignment, this function can only return one if a match is found and zero otherwise. Eventually, the $Q$-loss function is defined as:

$$\Delta_Q(y^o, y) = 1 - \frac{\sum_{k=1}^{\lvert y \rvert} \delta(y^o, y_k)}{N}.$$  

(4.7)

where $N$ is the number of index pairs in the ground-truth alignment. The $Q$-loss is a recall-like measure which returns zero if the predicted alignment contains all the index pairs of the ground truth, and proportionally up to one in case of missed pairs. Function $1 - \Delta_Q(y^o, y)$ is therefore a measurement of accuracy, and we refer to it as $Q$-accuracy hereafter.

The key step of the training of structural SVM is the loss-augmented inference in (3.11). Equation (4.7) shows that, like the score function, also the $Q$-loss decomposes over the individual states of the predicted alignment, $y_k, k = 1 \ldots \lvert y \rvert$. Thus, the loss-augmented inference takes the form:

$$\arg\max_{y \in Y} \sum_{k=1}^{\lvert y \rvert} \left[ w^\top \phi(s, t, y_k) - \delta(y^o, y_k) \right]$$

(4.8)

which can, again, be computed by the same algorithm used for the inference by adding the loss to the emission scores.

Another useful loss function is the $Q_4$-loss. This is a more lenient loss function that counts a match as correct even if the paired indices in the prediction are shifted by $\pm 2$ compared to those in the ground truth (i.e., “close enough”). In other terms, given $y_k = (i, j)$, a
match with \((m, n)\) in the ground truth is stated if \( |i - m| \leq 2 \) and \( |j - n| \leq 2 \). The \(Q_4\)-loss is, too, decomposable, and we define the \(Q_4\)-accuracy as \(1 - Q_4\)-loss.

In the experiments, we perform training using (4.8) and report results in terms of both \(Q\)-accuracy and \(Q_4\)-accuracy. In addition, during the annotation of the training set we have annotated the ground-truth alignments only for “key” frames that we have been able to pair with high confidence (e.g., apex phases of movements). The accuracy is measured only against such key frame pairs. Figure 4.2 summarizes the two main steps of the proposed approach.

4.3.4 Interpolation of the Ground-Truth Alignments Based on Key-Frames

Interpolation is a method of constructing new data points within the range of a discrete set of known data points. As we stated in the opening of this chapter, annotating complete ground-truth alignments of video pairs is very time-consuming. In addition, it can also prove ambiguous and/or subjective. However, in the case of human actions, one could exploit the fact that most actions have “key” points (such as the passing of the crossbar in pole vault or the reaching of the apex in weightlifting). Instead of annotating every frame pair along the two sequences, pairing only their key-frames promises to be more practical and reliable. An ensuing step of interpolation should then be able to provide an adequate, complete ground-truth alignment. We describe the interpolation procedure with a simple example hereafter.

Let us have two sequences, \(s\) and \(t\), and the ground-truth alignment of their key-frames, \(G(s, t)\), containing \(P_k\) pairs. For instance, in the example in figure 4.3, \((1, 1), (5, 6), (9, 11), (12, 15)\). We then divide \(G(s, t)\) into \(P_k - 1\) pairs of sub-sequences. The first frame of each sub-sequence is the first unutilised frame in the sequence, while the last frame in the sub-sequence is the next available key-frame. For instance, in the example, the pairs of sub-sequences are: \{\((1 : 5), (1 : 6)\)\}, \{\((6 : 9), (7 : 11)\)\}, and \{\((10 : 12), (12 : 15)\)\}. For each pair of sub-sequences, interpolating gap indexes are then added based on the following criteria:
Figure 4.3: Example of the interpolations on key-frames ground-truth alignments on the pairs of (1,1), (5,6), (9,11), (12,15) from UCF101 dataset.
\[ GAP = \begin{cases} 
  \text{yes} & \text{if } D(s_i, t_j) > \eta D(s^k_i, t^k_j) \\
  \text{no} & \text{if } D(s_i, t_j) \leq \eta D(s^k_i, t^k_j) \\
  \text{no} & \text{if } \text{Len}(s) = \text{Len}(t) 
\end{cases} \] (4.9)

where \( \eta \) is a threshold that was set to 0.8 in our experiments and \( D \) is the dissimilarity function. Algorithm 4 shows the main steps of the gap interpolation procedure.

**Algorithm 4:** Interpolation of ground-truth alignments based on key-frames: main steps.

**Input:** Ground-truth alignment of the key-frames \( G^* \); \( s^k \) and \( t^k \) note the \( k \)-th pair of key-frames in sequences \( S \) and \( T \).

Postprocessed ground-truth alignment \( G = \emptyset \)

\[ s_{p rev} = 1, \quad t_{p rev} = 1, \]

foreach \((s_i, t_j) \in G^*_t\)

\[ s \leftarrow S(s_{p rev}, s_i) \quad t \leftarrow T(t_{p rev}, t_j) \]

if \( \|s\| = \|t\| \) then

\[ G \leftarrow G \cup (s, t); \]

else

foreach \(i = 1 \ldots \|s\|, j = 1 \ldots \|t\|\)

if \( \|s\| = \|t\| \) then

\[ G \leftarrow G \cup (s, t); \]

break;

end

if \( GAP(s_i, t_j) \) then

if \( \|s\| < \|t\| \) then

\[ s_{location} \leftarrow \text{gap} \]

end

else

\[ t_{location} \leftarrow \text{gap} \]

end

end

end

**Output:** Interpolated ground-truth alignment \( Y \)
4.4 Experimental Results

In order to evaluate the performance of the proposed model, we have carried out experiments against the baseline model, DTW [1], and a state-of-the-art algorithm, GCTW [13], over selected actions from two action video datasets. The first experiment has aligned instances of “body weight squats” from the UCF101 dataset. Since the proposed model can align generic data streams, in the second experiment we have aligned instances of 3D joint sequences from action “stand up” in the MSR Daily Activity 3D dataset [53].

As software for structural SVM, we have used the \textit{SVMstruct} package of Joachims [15]. For training, we have set parameters $C$ to 10 and $\epsilon$ to 0.01 from a preliminary analysis, noting very limited sensitivity. As software for GCTW and DTW, we have used the package of Zhou [13], using the author’s values for the parameters. The accuracy is reported in terms of both $Q$-accuracy and $Q_4$-accuracy (see Section 4.3.3).

4.4.1 UCF101 Dataset

UCF101 is a large dataset of videos collected from YouTube with 101 action categories. For this experiment, we have chosen action “body weight squats” which was performed by 23 people under different scenes and viewpoints. As frame measurements, we have extracted the same features as the previous dataset (dense STIPs, bag-of-words encoded). We have then manually aligned the key frames of 253 video pairs and split them into 126 pairs for training and 127 for testing.

Table 4.2 shows the alignment results for the “body weight squats” action. The proposed model with the cosine distance has outperformed DTW in terms of $Q$-accuracy by 15.8 percentage points, and GCTW by 10.3 percentage points. The differences remain remarkable also in terms of $Q_4$-accuracy. Like for the Olympic Sports dataset, the cosine distance has achieved the highest accuracy, confirming that it is the most suitable for bag-of-words features. However, the proposed model has, again, outperformed DTW and GCTW also with the other distances. To further illustrate the results, Figure 4.4 shows a visual example of the alignment of two videos with the proposed model. The correspondence between key frames seems remarkably accurate. To give an idea of the computational times, training
Chapter 4. Extended Hidden Markov Support Vector Machine and its Application for the Alignment of Human Actions

65

Table 4.2: Experimental results for action “body weight squats” from the UCF101 dataset.

<table>
<thead>
<tr>
<th></th>
<th>GCTW</th>
<th>DTW</th>
<th>Proposed approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-accuracy</td>
<td>47.2%</td>
<td>41.7%</td>
<td>57.5%</td>
</tr>
<tr>
<td>Q₄-accuracy</td>
<td>69.3%</td>
<td>66.2%</td>
<td>81.1%</td>
</tr>
</tbody>
</table>

Table 4.3: Experimental results for action “stand up” from the MSR Daily Activity 3D dataset.

<table>
<thead>
<tr>
<th></th>
<th>GCTW</th>
<th>DTW</th>
<th>Proposed approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-accuracy</td>
<td>65.2%</td>
<td>43.2%</td>
<td>67.8%</td>
</tr>
<tr>
<td>Q₄-accuracy</td>
<td>86.9%</td>
<td>69.6%</td>
<td>94.3%</td>
</tr>
</tbody>
</table>

the proposed model on this training set has taken only 68.74 seconds on a PC with a 2.80 GHz i7-7600U CPU and 16 GB of RAM.

4.4.2 MSR Daily Activity 3D Dataset

This dataset is a popular activity dataset captured by a Microsoft Kinect device. It contains 16 activities of daily living performed by 10 subjects in two different poses: a standing position and a sitting position. The data included in this dataset comprise three channels: RGB, depth and “skeleton”. The skeleton channel encodes 20 joints per frame as displayed in Table 4.4.a. Each joint is represented by its estimated real world coordinates (x, y, z) and its screen coordinates plus depth (u, v, depth). The skeleton information is derived from the depth channel and is therefore naturally synchronized with it. The RGB channel runs instead on a separate thread and can be slightly shifted in time. For annotation, we have decided to annotate the alignment of the key frames using first the RGB channel, and then correct possible synchronization errors using the skeleton sequences. Table 4.4 shows an example of skeleton sequence superimposed, respectively, to its depth and RGB frames (rendered in gray-levels for greater clarity).

For this experiment, we have chosen action “stand up”, manually annotating 45 pairs and using 22 as training set and 23 as test set. The 3D joint positions in the skeleton data were first preprocessed with the invariant features from [53]. Table 4.3 shows the alignment
Figure 4.4: Example of alignment obtained with the proposed approach for two videos of action “body weight squats” from UCF101.
results for the “stand up” action. The proposed model with the Euclidean distance has outperformed DTW and GCTW, respectively, by 33.2 and 11.2 percentage points of $Q$-accuracy and 26.1 and 8.8 percentage points of $Q_4$-accuracy. Again, the proposed model has reported higher accuracies with all the distances. For a visual assessment, Table 4.6 shows an example of predicted skeletons from the proposed model and the compared algorithms, with arrows pointing to noticeable inaccuracies. The skeletons predicted by the proposed model look the closest to the ground truth.

4.5 Conclusion

In this chapter, we have presented an extended hidden Markov model (EHMM-SSVM) for alignment of sequence pairs that can learn an optimal, minimum-risk model from training sets of key-frame ground-truth alignments. The approach integrates an extended Hidden Markov Model to provide the sequences alignment and the structural SVM framework to optimally train its parameters. The main contributions of this chapter have been: a) a generalized linear score function suitable to score alignments paths; b) various distance functions to assess the dissimilarity of any two video frames; c) two loss functions that gradedly assess the quality of a predicted alignment against a given ground truth; d) an
interpolation procedure that allows us to obtain an adequate, complete ground-truth alignment from a partial alignment of only key-frames - this procedure makes the preparation of the training data quicker and more reliable. In addition, since both the score and loss functions are decomposable frame-by-frame, we have been able to retain an efficient, dynamic programming approach for the loss-augmented inference required by structural SVM for training.

The proposed model has been tested over selected actions from two, popular human action datasets against a baseline algorithm (DTW) and a state-of-the-art algorithm (GCTW [12, 13]). In the experiments, we have used diversified measurements to characterize the frames of the various datasets, including spatio-temporal descriptors and skeletons. The experimental results can be regarded as very encouraging since the proposed model has outperformed the compared algorithms by a large margin in all experiments, both for the stricter and the more lenient alignment measures.

As a final comment, the proposed method also enjoys the ability to act as a generalized distance measurement between two action sequences, making it suitable for action classification with minimum-distance classifiers. In Chapter 5, in fact, we will extensively explore the usefulness of the proposed EHMM-SSVM for tasks of action classification.
Chapter 5

Dissimilarity-Based Action Recognition with an Extended Hidden Markov Support Vector Machine

5.1 Introduction

The rapid growth in the availability of human action videos is pushing the need for tools that can automatically classify actions in applications such as sports video analysis, automatic surveillance, social media tagging and others. The field of automatic action recognition is constantly focussed on the design of more effective features and classifiers [56]. However, the problem remains challenging to date due to the many degrees of variations under which human actions can be performed and recorded.

The goal of this chapter is to explore action recognition by a dissimilarity-based approach where classification is performed in terms of minimum distance from labelled templates. Our motivation comes from the studies on dissimilarity-based classification that have all reported remarkable accuracy [16–20]. Dissimilarity-based classification does not represent each object by a feature vector: rather, it provides a distance function that can quantify
Chapter 5. *Dissimilarity-Based Action Recognition with an Extended Hidden Markov Support Vector Machine*

Figure 5.1: Left: Example of frame-by-frame distances. Right: Example of dynamic time warping.

The dissimilarity between any given object pair. Given a set of class-labelled templates, classification can thereby be provided by the use of a minimum-distance classifier (e.g., \(k\)-NN). Dissimilarity-based classification has proved very successful for the classification of non-vector objects such as graphs and strings [16–18] and seems promising also for the classification of complex sequential data such as human actions.

The crux of dissimilarity-based classification is the choice of an effective distance function. In the case of action videos, a basic idea could be to measure the distance between any two given videos simply as the sum of the distances between their frame pairs in appearance order. However, such a distance is too crude since it does not take into account local disalignments - see the left diagram in Fig. 5.1. For this reason, alignment algorithms such as dynamic time warping (DTW) [1], generalized canonical time warping (GCTW) [12, 13] and many others have been used to more suitably measure the distance between two videos. These algorithms are generally more effective since they compensate for temporal distortions - see the right diagram in Fig. 5.1. However, they are typically based on fixed cost models and they are difficult to adapt to specific applications.

Amongst alignment models, the Extended Hidden Markov Support Vector Machine (EHMM-SSVM) that we presented in chapter 4 is a more sophisticated alignment algorithm that has proved capable of remarkable accuracy, outperforming established algorithms such as DTW [1] and GCTW [12, 13] on various alignment metrics. The EHMM-SSVM provides major advantages over conventional alignment approaches: 1) an ability to learn an optimal cost model from any set of manually-aligned video pairs; 2) a maximum-margin objective that has a strong reputation for empirical accuracy; 3) the possibility to choose arbitrary loss functions for tuning the model to specific type of data; and 4) a customisable kernel distance between frame pairs for implementing nonlinear cost models (function \(\mathcal{K}\) in Equation 5.1).
In this chapter, we will explore the use of the EHMM-SSVM as underlying distance measure in tasks of human action classification. We will experiment with two types of classification approaches: minimum-distance classification and distance-based feature vectors. In the former approach, classification is performed in terms of minimum distance from a set of class-labelled training samples. In the latter, the underlying distance is used to form a feature vector of distances from the training samples which is used as input with any conventional classifier. As minimum-distance classifier, we have adopted the \( k \)-nearest neighbours (\( k \)-NN) [21]. To curb its computational complexity, we have applied a prototype selection technique that selects a number of template samples, or “prototypes”, from each class to abate the overall number of comparisons. For the second approach, we have computed the distances between each input sequence and the prototypes of every class, and used it as feature vector with a multiclass SVM classifier. We have tested the proposed approach in a set of experiments on action recognition over the popular KTH [57] and Olympic Sport [58] datasets. The experimental results show that the proposed approach has been able to outperform many existing approaches in terms of classification accuracy and rank closely to the state of the art.

### 5.2 EHMM-SSVM distance

Given the frame sequences of two videos, \( s = \{s_1, ..., s_i, ..., s_{L_s}\} \) and \( t = \{t_1, ..., t_j, ..., t_{L_t}\} \), an “alignment path”, \( y \), is a sequence of symbols that pairs frames from \( s \) and \( t \). The symbols are of three types: \( M \) (“match”), \( S \) (“insert a gap on sequence \( s \)”), and \( T \) (“insert a gap on sequence \( t \)”), with the following meaning: assuming that \( i \) and \( j \) are current
indices over sequences $s$ and $t$, respectively, 1) symbol $M$ pairs frames $s_i$ and $t_j$, and then increments both indices; 2) symbol $S$ pairs no frames and only increments index $j$; and, likewise, 3) symbol $T$ pairs no frames and only increments index $i$. To illustrate the alignment, Fig. 5.3 shows a diagram with two input sequences and an alignment path, including matched frames and inserted gaps; Fig. 5.2 shows actual examples of alignments for actions from the Olympic Sports dataset.

The EHMM-SSVM model 4 is a probabilistic model that defines a joint probability, $p(s, t, y)$, for sequences $s, t$ and their alignment path, $y$. This joint probability is chosen from the class of the exponential family of distributions, $p(s, t, y) \propto \exp(w^\top \psi(s, t, y))$, where $w$ notes a parameter vector and $\psi$ a suitable feature function. With this parametrization, an EHMM-SSVM can be trained by leveraging maximum-margin approaches that have gained a strong reputation for accuracy [15].

Like in a conventional hidden Markov model, the joint probability of an EHMM-SSVM conveniently factorizes into a set of transition and emission probabilities (Fig. 5.3 shows the model as a graphical model). Accordingly, parameter vector $w$ divides in two parts: transition parameters, $w^{tr}$, and emission parameters, $w^{em}$. The overall score function, $\text{dist}(s, t) = w^\top \psi(s, t, y)$, is written as:
where $I$ is the indicator function and indices $i$ and $j$, initially set to 1, are post-incremented according to the value of label $y_k$. Function $K$ is a generic distance function or kernel function which accounts for the dissimilarity between any two frames of $s$ and $t$.

Once a model and two input sequences are given, the optimal alignment $\bar{y} = \arg\max_y w^T \psi(s, t, y)$ can be computed by an efficient dynamic programming algorithm of linear complexity akin to the Viterbi algorithm [25]. In the following, we use the score of this optimal path as the inverse distance between the two input sequences. For training the EHMM-SSVM model, we have used structural SVM [15] over a set of manually-aligned video pairs in chapter 4.

## 5.3 Prototype selection with EHMM-SSVM

The main drawback of nonparametric dissimilarity-based classifiers such as $k$-NN is their computational complexity at run time: in principle, each test sample should be compared with every sample in the training set. While mitigation techniques such as the use of the triangle inequality and k-d trees can be exploited, the problem remains intrinsically complex, especially for large datasets. An alternative to the full run-time search is offered by prototype selection: in this case, the training set is replaced by a subset of representative prototypes, making the search substantially faster. In addition to reducing the run-time complexity, prototype selection typically achieves a comparable or even higher classification accuracy than the full search thanks to the removal of noisy and redundant samples [26].

The selection of the best prototypes can be performed according to a number of different criteria, including uniform distribution, centrality in the class, and others [17]. For this work, we have decided to adopt $KCcentres$ that selects prototypes that well reflect the sample distribution inside each class [27]. For each class, $KCcentres$ chooses $L$ prototypes
with these steps: 1) randomly pick an initial set of prototypes, \( P = \{p_1, \ldots, p_L\} \); 2) partition all the class’ samples into \( L \) subsets, \( J_1 = \{p_1\}, \ldots, J_L = \{p_L\} \), based on their closest prototype; 3) for each \( J_l, l = 1, 2, \ldots, L \), find its most central element (the element whose maximum distance to all other elements is minimum); 4) replace the prototypes in \( P \) with the most central elements. In step 3, EHMM-SSVM distance will be used for the comparision and produces a score for each pair of inputs. Eventually, iterate steps 2-4 until convergence or a maximum number of iterations is reached.

### 5.4 Classification

For classification, we have used two different approaches: 1) minimum-distance classification with a \( k \)-NN classifier and 2) a distance-based feature vector with a DAGSVM classifier. In the first approach, the \( k \)-NN classifier predicts the class of a test sample by the majority class of its \( k \) nearest neighbours. Despite its simplicity, the \( k \)-NN classifier has reported state-of-the-art accuracy over a number of benchmarks [21]. Before running \( k \)-NN, we have selected \( L \) prototypes per class and replaced the training set with the prototypes. This reduction abates the run time by the ratio between the size of these two sets. However, the prediction of a \( k \)-NN classifier is ambiguous whenever the majority class is at a parity. In this case, rather than picking a class using an arbitrary criterion, we have resorted to a “refinement classifier” to disambiguate the choice. As refinement classifier, we have used a multiclass DAGSVM with a standard bag-of-words (BoW) representation of the video as input. This idea was inspired by [19], although our implementation is simpler and faster. The overall classification procedure is summarised in Algorithm 5.

In the second approach, we have first computed a feature vector for each video consisting of the distances between the video itself and all the prototypes from all classes. Then, we have performed action classification using these feature vectors and a multiclass DAGSVM classifier [59]. A DAGSVM classifier (reading as “directed acyclic graph” SVM) is an improvement over the popular “one-vs-one” and “one-vs-all” SVM [60] that, by design, is capable of preventing classification parities. In one-vs-one and one-vs-all SVM, mutliclass classification is performed as a set of binary classifications that can often lead to classification ambiguities (samples than are classified into more than one class, or none). DAGSVM
Algorithm 5: k-NN classifier with EHMM-SSVM distances.

**Input:** Class set $\mathcal{C} = \{c_1, ..., c_n\}$; Test set $T = \{t_1, ..., t_l\}$; Prototype set $P = \{P_{c_1}, ..., P_{c_n}\}$

Allocate $l$ result sets $r = (r_1, ..., r_l)$

**foreach** $i \in 1 \ldots l$ **do**

Compute the EHMM-SSVM distances against $P$:

$D(t_i, P) = \{d(t_i, P_{c_1}),...,d(t_i, P_{c_n})\}$

**if** there is a majority class, $c$ **then**

$c \rightarrow r_i$

**end**

**else**

Apply DAGSVM and label $t_i$ to class $\hat{c}$

$\hat{c} \rightarrow r_i$

**end**

**end**

**Output:** result sets $r$

---

prevents these cases by organising the set of binary classifications as a decision tree: in each node of the tree, a single binary classification is performed to exclude one of the classes in turn. When the sample eventually reaches a leaf of the tree, its class assignment is unique. DAGSVM is also efficient since it requires only $M - 1$ binary inferences for classification over a set of $M$ classes. As an example of the feature vector, Table 5.1 shows the EHMM-SSVM distances between a sample of the “walking” class in the KTH dataset and 10 prototypes from each of the dataset’s 6 classes. To further illustrate the alignment distance, Figure 5.5 shows the EHMM-SSVM distances between every sample pair in the same dataset (2,391 instances) as a grey-scale matrix. The overall classification procedure is summarised in Algorithm 6.

Algorithm 6: DAGSVM classifier with EHMM-SSVM distances.

**Input:** Class set $\mathcal{C} = \{c_1, ..., c_n\}$; Test set $T = \{t_1, ..., t_l\}$; Prototype set $P = \{P_{c_1}, ..., P_{c_n}\}$

Allocate $l$ result sets $r = (r_1, ..., r_l)$

**foreach** $i \in 1 \ldots l$ **in** $T$ **do**

Transform $t_i \rightarrow t_i'$ against $P$:

$t_i' = \{d(t_i, P_{c_1}),...,d(t_i, P_{c_n})\} \in \mathbb{R}^{n \times P}$

Apply DAGSVM and label $t_i'$ to class $c$

$c \rightarrow r_i$

**end**

**Output:** result sets $r$
Figure 5.4: Sample actions in the Olympic dataset (top four rows) and KTH dataset (bottom row).

Table 5.1: Example of EHMM-SSVM distances between a walking sample in the KTH dataset and prototypes from the various classes. Darker colours denote higher similarity.
5.5 Experiments and Discussion

This section compares the performance of EHMM-SSVM distance-based classification against state-of-the-art methods over four experiments. To prepare the measurements for the experiments, we have first extracted dense feature descriptors from each frame of all the video sequences by using the STIP extractor of [57]. We have then computed a bag-of-words with 1,000 bins for each frame using the VLFeat library [61]. After that, we have trained a EHMM-SSVM distance model for each class on manually-aligned sequence pairs (ground-truth alignments) from that class. We have not trained cross-class models, expecting that a trained class model would return the highest scores for test sequences from the same class. For the annotation of the ground-truth alignments, we have selected and matched “key frames” (i.e., apexes of actions) from the paired sequences. After the EHMM-SSVM training, prototype selection has been performed on each class using the respective EHMM-SSVM distance to select \( L \) sequences as prototypes.

The experiments have been carried out over the KTH [57] and Olympic Sports [58] datasets. KTH is a video dataset of 6 action classes staged by 25 actors in various indoor and outdoor scenarios for a total of 2,391 action instances. The Olympic Sports dataset is a sport action dataset containing 16 action classes and a total of 800 action instances. This dataset is more challenging than KTH since its samples are real videos shot under a variety of viewing conditions and from different, unknown cameras.

5.5.1 Results with the \( k \)-NN classifier on the KTH dataset

For this experiment, we have used the standard training and test sets provided by the dataset’s authors. To train the EHMM-SVMM models, we have selected 16 training sequences per class and generated 120 (i.e., \( 16 \times 15/2 \)) manually-aligned pairs using 6 key frames for each sequence. The experiments have been carried out with 1, 5, and 10 nearest neighbours (1-NN, 5-NN and 10-NN) and the number of prototypes per class, \( L \), has been set to twice the number of the nearest neighbours. Table 5.2 shows that the accuracy from 10-NN has proved the highest, outperforming the results retrieved from the literature (including the method from Niebles et al. [58] by more than 23 percentage points).
The lower accuracy for classes “running” and “jogging” has been likely due to their higher cross-similarity that has made it difficult for EHMM-SVMM to yield reliable alignments. Conversely, neatly characterised actions such as “boxing” and “waving” have been recognised with 100% accuracy. Figure 5.5 clearly shows some degree of confusion between classes “walking”, “jogging” and “running”, which explains the lower accuracy for these classes.
Chapter 5. Dissimilarity-Based Action Recognition with an Extended Hidden Markov Support Vector Machine

Table 5.2: Accuracy with the k-NN classifier on the KTH dataset.

<table>
<thead>
<tr>
<th>Action</th>
<th>1-NN</th>
<th>5-NN</th>
<th>10-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>walking</td>
<td>83.3%</td>
<td>97.9%</td>
<td>99.3%</td>
</tr>
<tr>
<td>running</td>
<td>72.2%</td>
<td>84.0%</td>
<td>86.8%</td>
</tr>
<tr>
<td>jogging</td>
<td>72.9%</td>
<td>86.1%</td>
<td>88.2%</td>
</tr>
<tr>
<td>waving</td>
<td>86.1%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>clapping</td>
<td>85.4%</td>
<td>98.6%</td>
<td>98.6%</td>
</tr>
<tr>
<td>boxing</td>
<td>86.8%</td>
<td>99.3%</td>
<td>100%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>95.5%</td>
</tr>
<tr>
<td>Niebles et al. [58]</td>
<td>71.9%</td>
</tr>
<tr>
<td>Laptev et al. [62]</td>
<td>91.8%</td>
</tr>
<tr>
<td>Wong et al. [63]</td>
<td>86.7%</td>
</tr>
<tr>
<td>Kim et al. [64]</td>
<td>95.3%</td>
</tr>
<tr>
<td>Wang et al. [65]</td>
<td>92.1%</td>
</tr>
</tbody>
</table>

5.5.2 Results with the k-NN classifier on the Olympic Sports dataset

For the second dataset, we have used the same training and test splits of [58] and performed EHMM-SSVM training with the same procedure of the first dataset. The experiments have been carried out with 5-NN and 10-NN (1-NN did not seem promising, given its limited performance on KTH and the more challenging videos). Table 5.3 shows that 10-NN has, again, achieved higher accuracy than 5-NN. Its per-class accuracies are comparable with those from a state-of-the-art classifier (Jain et al. [66]): lower in 8 cases and on average, but higher in another 8 cases. The lowest accuracies were obtained for classes “discus-throw” and “shot-put” which were often confused because of their similar appearance; class “shot-put” is also very challenging in its own right because it is performed in a variety of styles (rotational, backsliding etc). Similar misclassifications have also occurred between classes "diving-platform" and "diving-springboard". Our approach has, again, achieved the highest accuracies for distinctive actions such as “clean-and-jerk” weightlifting and “pole-vault”. Overall, the average accuracy of the proposed method has proved 10.7 percentage points higher than the baseline from Niebles et al. [58] and has ranked closely to that of Jain et al. [66].

5.5.3 Results with the DAGSVM classifier on both datasets

For the experiments with the distance-based feature vector and the DAGSVM classifier we have used similar settings as with the k-NN classifier. For the KTH dataset, Table 5.4 shows that the accuracy with 10 prototypes has proved higher than the results retrieved from the literature (including the method from Niebles et al. [58] by approximately 24 percentage points), and also slightly higher than with the k-NN approach (0.3 percentage points). However, the same classes (i.e., “running”, “jogging”) have reported lower accuracy
Table 5.3: Accuracy in the second experiment on the Olympic Sports dataset.

<table>
<thead>
<tr>
<th>Sport class</th>
<th>Ours</th>
<th>Niebles et al.</th>
<th>Jain et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5-NN</td>
<td>10-NN</td>
<td>[58]</td>
</tr>
<tr>
<td>high-jump</td>
<td>79.2%</td>
<td>82.5%</td>
<td>68.9%</td>
</tr>
<tr>
<td>long-jump</td>
<td>82.7%</td>
<td>85.2%</td>
<td>74.8%</td>
</tr>
<tr>
<td>triple-jump</td>
<td><strong>83.5%</strong></td>
<td>85.7%</td>
<td>52.3%</td>
</tr>
<tr>
<td>pole-vault</td>
<td>84.3%</td>
<td><strong>88.1%</strong></td>
<td>82.0%</td>
</tr>
<tr>
<td>vault</td>
<td>82.2%</td>
<td><strong>86.5%</strong></td>
<td>86.1%</td>
</tr>
<tr>
<td>shot-put</td>
<td>69.3%</td>
<td>71.4%</td>
<td>62.1%</td>
</tr>
<tr>
<td>snatch</td>
<td>82.8%</td>
<td><strong>86.1%</strong></td>
<td>69.2%</td>
</tr>
<tr>
<td>clean-jerk</td>
<td>87.7%</td>
<td><strong>90.1%</strong></td>
<td>84.1%</td>
</tr>
<tr>
<td>javelin-throw</td>
<td>82.6%</td>
<td><strong>85.0%</strong></td>
<td>74.6%</td>
</tr>
<tr>
<td>hammer-throw</td>
<td>85.1%</td>
<td><strong>87.5%</strong></td>
<td>77.5%</td>
</tr>
<tr>
<td>discus-throw</td>
<td>70.1%</td>
<td>73.3%</td>
<td>58.5%</td>
</tr>
<tr>
<td>diving-platform</td>
<td>73.1%</td>
<td>78.9%</td>
<td><strong>87.2%</strong></td>
</tr>
<tr>
<td>div. springboard</td>
<td>70.6%</td>
<td>74.3%</td>
<td>77.2%</td>
</tr>
<tr>
<td>basketball</td>
<td>78.3%</td>
<td>81.6%</td>
<td>77.9%</td>
</tr>
<tr>
<td>bowling</td>
<td>82.8%</td>
<td>86.2%</td>
<td>72.7%</td>
</tr>
<tr>
<td>tennis-serve</td>
<td>77.2%</td>
<td>81.8%</td>
<td>49.1%</td>
</tr>
</tbody>
</table>

Table 5.4: Accuracy with the DAGSVM classifier on the KTH dataset.

<table>
<thead>
<tr>
<th>Action</th>
<th>1-PT</th>
<th>5-PT</th>
<th>10-PT</th>
<th>Algorithm</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>walking</td>
<td>81.3%</td>
<td>98.9%</td>
<td>99.3%</td>
<td>Ours</td>
<td><strong>95.8%</strong></td>
</tr>
<tr>
<td>running</td>
<td>71.9%</td>
<td>86.2%</td>
<td>87.7%</td>
<td>Niebles et al. [58]</td>
<td>71.9%</td>
</tr>
<tr>
<td>jogging</td>
<td>72.3%</td>
<td>88.0%</td>
<td>89.1%</td>
<td>Laptev et al. [62]</td>
<td>91.8%</td>
</tr>
<tr>
<td>waving</td>
<td>87.4%</td>
<td>100%</td>
<td>100%</td>
<td>Wong et al. [63]</td>
<td>86.7%</td>
</tr>
<tr>
<td>clapping</td>
<td>86.2%</td>
<td>98.6%</td>
<td>99.1%</td>
<td>Kim et al. [64]</td>
<td>95.3%</td>
</tr>
<tr>
<td>boxing</td>
<td>87.2%</td>
<td>99.3%</td>
<td>100%</td>
<td>Wang et al. [65]</td>
<td>92.1%</td>
</tr>
</tbody>
</table>

also with this classifier. This clearly shows that the performance of the classifier is closely tied to the discriminative capability of the underlying EHMM-SSVM distance.

For the Olympic Sports dataset, the accuracies reported in Table 5.5 seem interesting and generally comparable to those of the $k$-NN classifier. The average accuracy is, again, slightly higher (0.3 percentage points). The DAGSVM classifier has also achieved higher scores than a state-of-the-art classifier (Jain et al. [66]) in 9 cases (with class “bowling” in addition to those from the previous experiment). The lowest accuracies have again been for class pairs “discus-throw” and “shot-put”, and “diving-platform” and “diving-springboard", due to their evident similarity. The proposed approach has again achieved its highest accuracies for distinctive actions such as “clean-and-jerk” weightlifting and “pole-vault". Overall, the average accuracy of the proposed method has proved 11 percentage points higher than the baseline from Niebles et al. [58] and has ranked closely to that of Jain et al. [66].
Chapter 5. **Dissimilarity-Based Action Recognition with an Extended Hidden Markov Support Vector Machine**

In this chapter, we have presented a novel dissimilarity-based approach to action recognition in videos. The approach leverages the recently-proposed EHMM-SVM alignment algorithm which, for every two given videos, provides an alignment path and a similarity score. In our experiments, we have used the EHMM-SSVM similarity score as an inverse distance between the two input videos, and exploited it for action classification. We have proposed two distance-based methods for classification: 1) a $k$-NN classifier using the EHMM-SSVM distance; 2) a DAGSVM classifier using a EHMM-SSVM-based feature vector. Prior to applying the classifiers, we have run a step of prototype selection to select a set of prototypes for each class. For the $k$-NN classifier, we have replaced the training set with the prototypes’ set to abate the test-time computational complexity. For the DAGSVM classifier, we have used the prototypes to obtain a distance-based feature vector for each video. The experimental results over two popular action video datasets - KTH and Olympic Sports - have showed that:

- on the KTH dataset, the proposed approaches have achieved an accuracy that is 24 percentage points higher than the classifier from Niebles et al. [58] and higher than the results compiled from the literature;

<table>
<thead>
<tr>
<th>Sport class</th>
<th>Ours 5-PT</th>
<th>Ours 10-PT</th>
<th>Niebles et al. [58]</th>
<th>Jain et al. [66]</th>
</tr>
</thead>
<tbody>
<tr>
<td>high-jump</td>
<td>77.8%</td>
<td>81.7%</td>
<td>68.9%</td>
<td>84.9%</td>
</tr>
<tr>
<td>long-jump</td>
<td>83.1%</td>
<td>85.6%</td>
<td>74.8%</td>
<td>84.6%</td>
</tr>
<tr>
<td>triple-jump</td>
<td>84.1%</td>
<td>86.1%</td>
<td>52.3%</td>
<td>83.3%</td>
</tr>
<tr>
<td>pole-vault</td>
<td>83.5%</td>
<td>86.6%</td>
<td>82.0%</td>
<td>84.7%</td>
</tr>
<tr>
<td>vault</td>
<td>82.6%</td>
<td>86.7%</td>
<td>86.1%</td>
<td>82.6%</td>
</tr>
<tr>
<td>shot-put</td>
<td>68.9%</td>
<td>74.1%</td>
<td>62.1%</td>
<td>83.6%</td>
</tr>
<tr>
<td>snatch</td>
<td>84.2%</td>
<td>86.4%</td>
<td>69.2%</td>
<td>83.5%</td>
</tr>
<tr>
<td>clean-jerk</td>
<td>88.3%</td>
<td>90.5%</td>
<td>84.1%</td>
<td>86.6%</td>
</tr>
<tr>
<td>javelin-throw</td>
<td>82.0%</td>
<td>85.1%</td>
<td>74.6%</td>
<td>84.8%</td>
</tr>
<tr>
<td>hammer-throw</td>
<td>86.6%</td>
<td>88.3%</td>
<td>77.5%</td>
<td>86.4%</td>
</tr>
<tr>
<td>discus-throw</td>
<td>70.5%</td>
<td>74.6%</td>
<td>58.5%</td>
<td>86.7%</td>
</tr>
<tr>
<td>diving-platform</td>
<td>71.8%</td>
<td>76.4%</td>
<td>87.2%</td>
<td>86.5%</td>
</tr>
<tr>
<td>div. springboard</td>
<td>70.1%</td>
<td>72.8%</td>
<td>77.2%</td>
<td>86.4%</td>
</tr>
<tr>
<td>basketball</td>
<td>80.4%</td>
<td>82.8%</td>
<td>77.9%</td>
<td>88.6%</td>
</tr>
<tr>
<td>bowling</td>
<td>86.4%</td>
<td>88.6%</td>
<td>72.7%</td>
<td>88.3%</td>
</tr>
<tr>
<td>tennis-serve</td>
<td>77.9%</td>
<td>83.2%</td>
<td>49.1%</td>
<td>83.4%</td>
</tr>
</tbody>
</table>

**Table 5.5:** Accuracy with the DAGSVM classifier on the Olympic Sports dataset.

### 5.6 Conclusion

In this chapter, we have presented a novel dissimilarity-based approach to action recognition in videos. The approach leverages the recently-proposed EHMM-SVM alignment algorithm which, for every two given videos, provides an alignment path and a similarity score. In our experiments, we have used the EHMM-SSVM similarity score as an inverse distance between the two input videos, and exploited it for action classification. We have proposed two distance-based methods for classification: 1) a $k$-NN classifier using the EHMM-SSVM distance; 2) a DAGSVM classifier using a EHMM-SSVM-based feature vector. Prior to applying the classifiers, we have run a step of prototype selection to select a set of prototypes for each class. For the $k$-NN classifier, we have replaced the training set with the prototypes’ set to abate the test-time computational complexity. For the DAGSVM classifier, we have used the prototypes to obtain a distance-based feature vector for each video. The experimental results over two popular action video datasets - KTH and Olympic Sports - have showed that:
• on the Olympic Sports dataset, the proposed approaches have achieved an accuracy that is 11 percentage points higher than the classifier from Niebles et al. [58] and close to that of a state-of-the-art approach [66].

A further analysis of the per-class accuracy has shown that the proposed approach has tended to outperform the other classifiers on actions with more pronounced temporal stages. Given that the EHMM-SVMM distance is based on temporal alignment, this result is encouraging and indicates that the best application for the proposed approach are actions with neatly-outlined stages.
Chapter 6

Conclusion

The meaningful alignment of data sequences is a crucial requirement for the analyses of correlations of sequential data. A proper alignment does not only consist of the registration of the initial and ending frames; rather, it needs to reflect the main “stages” of each sequence and provide a dense set of pairings between their frames. *Alignment*, or *warping, algorithms* are designed to provide such alignments, typically between two sequences at a time, and often with one treated as the “reference template” and the other as test sequence.

In this thesis, we have presented a novel approach to the problem of sequence alignment - named extended hidden Markov support vector machine (EHMM-SSVM) - and showed its effectiveness for both video alignment and dissimilarity-based classification of human actions. In chapters 3, we have proposed the integration of the rigorous probabilistic formulation of the pair HMM with the effective parameter training of structural SVM. The proposed integration has included dedicated feature and loss functions suitable to achieve accurate video alignments. To retain the efficient training of structural SVM, we have also proposed a dynamic programming algorithm for loss-augmented inference with the dedicated loss functions. In chapter 4, we have extended our initial approach to a model called the EHMM-SSVM. The improvements include: a) a generalized linear score function suitable to score alignments paths; b) various distance functions to assess the dissimilarity of any two video frames; c) two loss functions that gradedly assess the quality of a predicted
alignment against a given ground truth; and d) the automated interpolation of ground-truth alignments from the partial annotation of key-frames only. Experimental results over four probing datasets have shown that the PHMM-SSVM and EHMM-SSVM have achieved higher alignment accuracy than both a standard dynamic programming algorithm (DTW) and a state-of-the-art algorithm (GCTW).

Eventually, in chapter 5 we have set to use our EHMM-SSVM algorithm as a generalised distance measure between action sequences, and combined it with minimum-distance classifiers for action classification. The EHMM-SVMM outputs a similarity score between its two input sequences which can be used as the (dis)similarity function in distance-based classifiers. We have explored two classification approaches: 1) a $k$-NN classifier using the EHMM-SSVM distance; 2) a DAGSVM classifier using an EHMM-SSVM-based feature vector. Because of the intrinsically high computational complexity of similarity-based classification, we have preliminarily applied prototype selection over the training data of each class to select a limited number of templates. The $k$-NN classifier then compares each test sample with all the prototypes of each class and assigns it the majority label amongst its nearest neighbours. At its turn, the DAGSVM classifier utilises as features the distances between the test sample and all the prototypes. The experimental results over the KTH and Olympic Sports datasets of human actions have shown promising accuracy and provided evidence that the EHMM-SVMM distance plays an important role in achieving this accuracy.

At the conclusion of this thesis, we feel confident to have offered a significant contribution to the problems of video alignment and alignment-based classification of human actions. However, as research is a world in continuous motion, we can already highlight extensions that would be worth exploring as future work. To name one, the integration of the proposed approaches with the burgeoning fields of deep architectures and reinforcement learning.
Bibliography


