

A GENERATIVE-DISCRIMINATIVE FRAMEWORK FOR
TIME-SERIES DATA CLASSIFICATION

KARIM T. ABOU-MOUSTAFA

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

DECEMBER 2003

© KARIM T. ABOU-MOUSTAFA, 2004



National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-90984-0

Our file Notre référence

ISBN: 0-612-90984-0

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

A Generative-Discriminative Framework for Time-Series Data Classification

Karim T. Abou-Moustafa

Discriminative models such as SVMs and MLPs are known for their good generalization in classification of static data. However, classification of time-series data using these models is still a very difficult task for two reasons: 1) discriminative models are unable to model the time variability in time-series data, and 2) time-series data usually have a variable length. Unlike discriminative models, generative models such as HMMs were able to overcome these problems and became the standard tool for modelling time-series data, but on the other hand, their classification performance is poor. This thesis targets the problem of poor performance of HMM-based classifiers. First, we study the effect of the structure on the performance of HMMs and see how the number of states and the topology can contribute to the classification performance. As a result, our investigation showed the topology has a stronger contribution to the classification performance than the number of states. Second, we propose a general two-stage framework that combines generative and discriminative models to reach a high performance in the classification of time-series data. In the first stage, HMMs are used to model the time-series data, then a fixed size score vector is extracted from this stage and used as the input to the discriminative model in the second stage. The framework showed a potential for combining generative and discriminative models for

time-series data classification and was able to achieve a recognition rate of 98.02%, with an increase of 3.83% over traditional HMM-based classifiers.

Acknowledgments

I would like to express my deep gratitude to my family for their love, encouragement and support. They always helped me and they will keep encouraging me to accomplish all my dreams.

My deep gratitude goes to my supervisor, Prof. Ching Y. Suen for giving me the chance, in the first place, to start my career in research. He opened the gate and let me in. As my supervisor, I would like to thank him for his continuous guidance, patience and support, as well as providing an excellent working environment at CENPARMI. My great appreciation and thanks go to my co-supervisor, Prof. Mohamed Cheriet, for the long way of valuable discussions and meetings during the development of my research. His excellent suggestions became the core of this thesis.

Special thanks to my colleagues at CENPARMI, in particular, Dr. Yousef Al-Ohali, Dr. Jincheol Kim, Andrea de Souza and Nedjem Eddine Ayat. I would also like to thank Dr. Jiangxiong Dong, Dr. Qizhi Xu, Jun Zhou (Garry) and Nenghong Fu for many good discussions we had together. Also, I would like to thank Beverley Abramovitz, Nicola Nobile and Halina Monkiewicz from Computer Science Department at Concordia University.

This page can never end without thanking my professors during my undergraduate studies, Prof. Magdy Saeb, Prof. Yasser El-Sonbaty, Prof. Ossama Badawy and Prof. Yasser Hanafi from the Department of Computer Engineering at the Arab Academy for Science and Technology, Alexandria, Egypt.

Contents

List of Figures	ix
List of Tables	xi
1 Introduction	1
2 State of the Art	7
2.1 Structure Optimization	8
2.1.1 The role of the priori knowledge	8
2.1.2 Optimizing the number of states and the topology	9
2.1.3 Optimizing the number of states only	10
2.1.4 Different approaches	11
2.2 Generative and Discriminative Models	13
3 Basics & Notations	16
3.1 Hidden Markov Models	16
3.1.1 Elements of HMMs	17
3.1.2 Observation likelihood, optimal state sequence and training . .	19
3.2 Graphical Models	21
3.2.1 Generative models	22
3.3 Support Vector Machines	25

3.3.1	SVM for linearly separable case	26
3.3.2	SVM for non-separable case	28
3.3.3	Nonlinear SVMs and kernels	29
3.3.4	SVM applied to multiclass classification	31
3.4	Notations	32
4	Notes on Hidden Markov Modelling	33
4.1	Hidden Markov Modelling Limitations	34
4.2	The Effect of the Structure	34
4.2.1	The topology factor	35
4.3	Parameter Estimation	38
4.3.1	Discussion	43
5	The Generative-Discriminative Framework	45
5.1	Bayes Decision Theory	46
5.2	Generative Vs Discriminative Models	47
5.3	The Proposed Framework	50
5.3.1	The likelihood score	51
6	Experimental Results	53
6.1	Feature Extraction	54
6.2	The Effect of the Structure on the Performance	55
6.2.1	HMM density type, initialization and codebook size	55
6.2.2	Studying the effect of number of states	55
6.2.3	Studying the effect of model topology	61
6.3	Experimental Results for the Proposed Framework	62
7	Conclusions and Future Work	72

List of Figures

1	A simple graphical model for two random variables.	21
2	Discrete hidden Markov model unrolled for 3 time slices.	23
3	An HMM with mixture of Gaussians output unrolled for 2 time slices.	23
4	Input-Output HMM (IOHMM) unrolled for 2 time slices.	24
5	Factorial HMM unrolled for 2 time slices.	24
6	Among the separating hyperplanes, SVM chooses the one which has the largest margin. The support vectors are circled.	27
7	Linear separating hyperplanes for the separable case. The support vectors are circled	28
8	Linear separating hyperplanes for the nonseparable case	29
9	The role of the mapping function ϕ in making the data more linearly separable	30
10	Convergence of Dobrushin's coefficient for 4 different topologies.	38
11	A block diagram of the proposed framework	50
12	Handwritten digits: samples of digits 1 are shown in the first row, while samples for digit 7 are shown in the second row	52
13	Samples from the training set of the MNIST data base.	53
14	The relation between performance and the number of states with dif- ferent code book sizes.	56

15	Performance comparison between models with equal (EQU) and varying (VAR) number of states with different codebook sizes.	58
16	Performance comparison between full ergodic and left-to-right models with different codebook sizes.	61
17	Confusions among digit 0 and other digits	65
18	Confusions among digit 1 and other digits	66
19	Confusions among digit 2 and other digits	66
20	Confusions among digit 3 and other digits	66
21	Confusions among digit 4 and other digits	67
22	Confusions among digit 5 and other digits	67
23	Confusions among digit 6 and other digits	68
24	Confusions among digit 7 and other digits	68
25	Confusions among digit 8 and other digits	69
26	Confusions among digit 9 and other digits	69

List of Tables

1	Summary of inner-product kernels	31
2	Comparing Learning in Generative and Discriminative models	49
3	Performance measures on the test set as the number of states and varying code book size increased.	57
4	Comparison between performances of EQU models and VAR models at each epoch on the test set using codebook size = 1024	59
5	Comparison between performances of EQU models and VAR models at each epoch on the test set using codebook size = 512	59
6	Comparison between performances of EQU models and VAR models at each epoch on the test set using codebook size = 256	60
7	The number of states of each model.	61
8	Performance measures obtained from full ergodic and left-to-right mod- els with different code book sizes.	62
9	HMM based classifier performance on the validation set and the test set with different codebooks	63
10	The proposed framework performance on the validation set and the test set with different codebooks in the modelling stage	64
11	Comparison of performance of the HMM based classifier and the pro- posed framework	64
12	Digit error rate for the proposed framework	64

13	Confusion matrix for the errors generated by the proposed framework	65
14	Comparison of performances of different classifiers on the MNIST database	71

List of Abbreviations

BIC	Bayesian Information Criterion
DIC	Discriminative Information Criterion
EM	Expectation Maximization algorithm
GDT	Generative-Discriminative Tradeoff
GM	Generative Models
HMM	Hidden Markov Models
MAP	Maximum A Posteriori
MCE	Minimum Classification Error
MDL	Minimum Description Length
MLP	Multi-layer Perceptrons
LVQ	Learning Vector Quantization
SVM	Support Vector Machine

List of Notations

A	Probability transition matrix
a_{ij}	Probability of moving from state i to state j
B	Observation probability matrix
$b_j(z_t)$	Probability of observing z at time t at state j
π_i	Probability of starting at state i
μ_{jk}	Mean vector of mixture k at state j
U_{jk}	Covariance matrix of mixture k at state j
$\mathcal{N}(\cdot, \mu, U)$	Gaussian distribution with mean μ_{jk} and covariance matrix U_{jk}
q_i	State i
Q_i	Random variable i
Z	A Time-Series (sequential) pattern
Z_0	An unknown input sequential pattern
X	An input vector, usually $X \in \mathbb{R}$
Y	The label of an input pattern
α	Parameter vector in the context of SVMs
θ	Parameter vector in the context of GMs or HMMs
$R(\cdot)$	Expected risk function
R_{emp}	Empirical expected risk
η	A <i>+ve</i> constant

h	A nonnegative integer, the VC dimension.
b	Bias term
ξ	Slack variable in the context of SVMs
\mathcal{H}	Hilbert space
C	Constant or error penalty in the context of SVMs
$K(.,.)$	Kernel function
S^h	Hypothesis structure, in the context of HMMs or GM
$I(.,.)$	Mutual Information criterion function
$l(.)$	MCE loss function
$L(.)$	Expected loss function

Chapter 1

Introduction

1.1 Motivation and Problem Description

A time-series pattern is a sequence of objects that appear in a certain order. Each object in the sequence appears at certain instant in time, and the occurrence of an object is dependent on the occurrences of previous objects in the sequence. If the objects are real valued vectors, i.e. $x \in \mathbb{R}$, then the sequence is said to be a continuous time-series pattern. If the objects are symbols from a finite dictionary, then the sequence is known as a discrete time-series pattern. Continuous time-series data can be obtained from modelling speech and handwritten words. Discrete time-series data can be obtained from modelling DNA sequences or quantization of continuous data. Time-series data of a single class can have different sequence lengths and different order and values for its object, which makes it difficult in handling this type of data.

For a classifier to process time-series data, first, it must be capable of modelling the time variability in order to recognize the sequence of objects, since two different orders of the same objects can lead to two different patterns, and second, it must be capable of dealing with the variable length of the patterns. Discriminative models for classification such as Multi layer perceptrons (MLP) neural networks and Support

Vector Machines (SVMs) that are well known for their good generalization in classification of static and fixed size patterns, to the best of our knowledge, can not be used to classify this type of data due to their inability to model the time variability and due to their inability to deal with variable length data.

HMMs were able to overcome these problems. Hidden Markov models or HMMs [BP66, Rab89, Ben99], are a class of stochastic processes that are capable of modelling the time and length variability of time-series data. They belong to a larger class of models known as generative models that are mainly used for data modelling. Generative models in general, and HMMs with no exception, are used to approximate the true density of the data, and hence the meaning of data modelling. Although, HMMs can be trained discriminately using Maximum Mutual Information (MMI) [BBdSM86] and Minimum Classification Error (MCE) [Bie02]. Since their first appearance in speech as tool for speech modelling [Jel72, Bak75], they became the standard tool for modelling and classifying time-series data.

Unlike discriminative models that are known for their good generalization in classification tasks, generative models in general have a low classification performance [NJ02]. This is due to several reasons that are discussed in detail in Section 5.2. In addition to the limitation of generative models in general, HMMs have also factors that affect its performance and hence might increase or decrease the classification performance of time-series data. These issues are addressed at length in Chapter 4.

In this thesis, we target the low performance of HMM-based classifiers for time-series data classification. In real world, many applications are related to this problem; time-series data classification occurs in applications such as speech recognition, handwritten word recognition, classifying biological sequences, weather prediction, stock prices prediction, and increasing the classification performance will certainly increase the performance of these systems.

1.2 Summary of Contributions

Our primary investigations targeted the problem of structure optimization to increase the performance of HMM-based classifiers. The investigation was based on theoretical results from the machine learning literature supported by our experiments that studied the effect of the number of states and the topology of the model, each separately, on the performance of HMMs. As it will be shown in Sections 4.1 and 4.2, the theoretical and the experimental results showed that the topology of the model has a stronger influence on the performance of HMMs than the number of states. Also, our research confirmed that the Bakis model, i.e, the left to right model with self state transitions is indeed a good model that can handle time-series data. In that sense, these results contradict the new approach that focused only on optimizing the number of states as was proposed in [BDM01, BMF03] and encourage, on the other hand, the optimization of the whole structure as Stolcke [SO92] and Brants [Bra96] algorithms do.

In the next investigation, we combined generative and discriminative models to achieve high performance time-series data classification. We proposed a new two-stage framework that combines the complementary features of generative and discriminative models to classify time-series data. The proposed framework is intuitively derived from the work of Jaakkola and Hausler [JH98]. Since generative models are able to model or describe the variable length time-series data and discriminative models are able to classify static fixed size vectors, then a static fixed size feature vector that contains enough information on the sequential pattern, can be extracted from the generative models and then use the discriminative model to classify it. In other words, after representing the time-series data using HMMs in the first stage, a feature vector can be extracted from the HMMs, the likelihood score, then SVMs or MLPs, in the second stage, can be used to classify this feature vector. In that sense, the role

of HMMs was reduced to data modelling only and the classification task is left for the discriminative model. The proposed framework was compared with traditional HMMs on the problem of recognition of unconstrained handwritten digits. A standard database, the MNIST [LeC] database, was used to benchmark the performance of both systems. The results of the simple prototype model used in the experiments showed the potential of the generative discriminative approach in general and the potential of our proposed combination method. Traditional HMMs were able to achieve 94.19%, while the framework was able to boost the results to 98.02%. The 3.83% increase encourages future investigations to improve the proposed framework to achieve better performance.

1.3 Thesis Organization

The thesis is organized in seven chapters. Readers familiar with these topics can jump directly to the core of the thesis in Chapters 4, 5 & 6. The chapters contents are as follows:

- Chapter 2 reviews previous and state of the art research work that were introduced in the literature for two main problems: 1) Increasing the performance of HMMs for classification tasks, and 2) Combining generative and discriminative models for classification problems. Increasing the performance of HMM-based classifiers depends mainly on increasing the discrimination between the models of the classifier. In the literature, two approaches are followed: 1) optimizing the model structure (the number of states and the topology, or 2) improving parameter estimation or learning algorithms. Section 2.1 reviews the structure optimization approach and presents algorithms that are dependent and independent from the application domain. To avoid redundancy and to keep the consistency of the context, the detailed review for parameter estimation algorithms is presented in Section 4.3 since it reviews, compares and analyzes

these algorithms and shows how they affect the performance of HMMs. Finally, Section 2.2 reviews the state of the art research in combining generative and discriminative models for classification problems.

- Chapter 3 introduces the basic theory of hidden Markov models, graphical models and support vector machines. Hidden Markov models form the core of this thesis so their basic theory comes in the first section. Section 2 reviews in brief the basics of graphical models in general and generative models in specific. Section 3 reviews the basic theory of support vector machines since they form the second stage of the proposed framework in this thesis. Finally, the notations that will be used through the thesis are illustrated in Section 4.
- Chapter 4 takes a deeper insight into hidden Markov modelling by focusing only on the representation capability of HMMs. The main issue covered in this chapter is understanding the factors that affect the modelling capability of HMMs and how it can affect the final performance of HMM-based classifiers. The modelling capability of HMMs is affected by two main factors: (I) the model structure, and (II) the parameter estimation or the learning (training) algorithm. The model structure can also be split to two factors: (a) the number of states, and (b) the topology, i.e. the connections between the states. In addition to the mentioned factors, there are other factors that arise from the nature of the model itself and have a direct effect on the modelling capability. Sections (1), (2) and (3) discuss all the above mentioned issues with more details.
- Chapter 5 introduces the new framework that combines the complementary features of HMMs and SVMs for classifying time-series data. Section (1) recalls Bayes decision theory, and Section (2) presents a thorough comparison between generative and discriminative models in general. Finally, section (3) introduces

the new framework.

- Chapter 6 illustrates all the experiments conducted in order to validate the investigation of this thesis. Section (1) introduces the features extracted from the images and how they were represented as time-series data. Section (2) shows the experiments that investigated the effect of the structure on the performance of HMM-based classifiers. The first part provides experimental results that support the theoretical results obtained in Section (4.2.1). The second part provides the experiments that investigated the effect of number of states on the performance of HMM-based classifiers. Finally, Section (3) shows the results obtained from the proposed framework when used for recognizing unconstrained handwritten digits from the MNIST database.
- Finally, Chapter 7 draws conclusions and describes future research directions.

Chapter 2

State of the Art

This chapter reviews previous and state of the art research work^s that were introduced in the literature for two main problems: 1) Increasing the performance of HMMs for classification tasks, and 2) Combining generative and discriminative models for classification problems. Increasing the performance of HMM-based classifiers depends mainly on increasing the discrimination between the models of the classifier. In the literature, two approaches are followed: 1) optimizing the model structure (the number of states and the topology, and 2) improving parameter estimation (training or learning) algorithms.

Section 2.1 reviews the structure optimization approach and presents algorithms that are dependent on and independent from the application domain. To avoid redundancy and to keep the consistency of the context, the detailed review for parameter estimation algorithms is presented in Section 4.3 since it reviews, compares and analyzes these algorithms and shows how they affect the performance of HMMs. Finally, Section 2.2 reviews the state of the art in combining generative and discriminative models for classification problems.

2.1 Structure Optimization

2.1.1 The role of the priori knowledge

The speech recognition community has introduced to the literature many state of the art techniques and results using HMMs. With a similar success but later in time, handwritten word recognition researchers, grasped many of the successful techniques of speech recognition and casted them in handwritten recognition problems. The main reason for state of the art results of speech recognition systems lies in the amount of a priori knowledge plugged in to the recognition system [Lee99, BR99]. A speech recognition system does not depend solely on HMMs and the simple Baum-Welch algorithm, but depends on two tools that provide the necessary a prior information. Word networks and language models are the true working horses of speech recognition system. They provide sufficient statistics and class priors that are plugged in every phase of the recognition system: segmentation, code book generation, training, recognition and post processing. Despite that these tools could be used in handwriting recognition, yet handwritten word recognition systems have a limited use for such information. However, other types of information is plugged in to handwritten recognition systems. El-Yacoubi et al. [EYGSS99] used prior information from a character segmentation process and used it only to build a special HMM structure. Although this information is not incorporated in training and neither in testing, this system achieved promising results and was the basic system for several recognition systems introduced later for large vocabulary word recognition [KRSG02] and address recognition [GSSG00]. Also, applications with limited vocabulary such as legal amount recognition, use a different sort of priori information [ABKP98]. Special purpose methods can not be applied generally to all other applications that use HMMs, but it can encourage researchers to search for a priori knowledge and incorporate it

in the recognition system.

2.1.2 Optimizing the number of states and the topology

Unlike special purpose methods that are used to increase the performance of HMM based classifiers, general methods that can optimize the structure from the data without a priori knowledge from the application domain are few. The well known Bakis model, or the left-to-right model [Bak76] was suggested to handle sequential data like speech. For many years this model was used in many systems, and our experiments showed the efficiency of this simple topology, however, other types of data may need different random topology that should be deduced from the data. The first work presented in the literature to optimize the structure of HMMs was by Stolcke and Omohundro [SO92]. The structure optimization in their work was based on incrementally learning the structure from the data, i.e. the structure is changed as new evidence is added to the model. The algorithm is based on an observation from the analysis of Bayes rule, that is, the model posterior $Pr(\lambda|\mathcal{Z})$ is proportional to the product of the model likelihood $Pr(\mathcal{Z}|\lambda)$ and the model prior $Pr(\lambda)$, where λ is the hidden Markov model and \mathcal{Z} is the training set for the model λ . As the model gets smaller or simpler, the likelihood will drop, and hence the prior has to compensate this decrease to increase the model posterior. The algorithm starts from a very large model, M_0 , and then iteratively merge some states and outputs some new models, M_1, M_2, \dots , where at each iteration, models are getting smaller. From these models, the algorithm search for a model that maximizes the posterior probability of the model. The algorithm stops when a decrease is found in the posterior. Indeed, the likelihood is the deriving force for this algorithm since it accounts for an overall increase in the posterior.

Followed by Stolcke [SO92], and using the same approach, Brants [Bra96] combines

state splitting with state merging. Determining either to split a state or to merge states is based on the number of samples available for each state. States with sufficient data are split, while those with very few samples are merged. The candidate state for a split operation is chosen based on the divergence of probability distribution that is based on the relative entropy. The algorithm starts with two sets of states, one for those states with sufficient samples Q_H , and the other for those with very few samples Q_L . Iteratively the algorithm applies merging for Q_L and splitting for Q_H .

The main advantage of both algorithms is their independence from the application domain and reliance on the available data. The drawback for the first algorithm and consequently for the second one also, is that they require lots of approximation and smoothing algorithms which can affect slightly the performance and therefore it is computationally demanding to ensure accuracy. Although Stolcke's algorithm appeared in 1992, yet the left-to-right model is preferred in many applications.

2.1.3 Optimizing the number of states only

Recently, Bicego et al. focused completely on optimizing the number of states using two methods, 1) Bisimulation algorithm [BDM01], and 2) Sequential Pruning [BMF03]. The first approach consists of eliminating syntactic redundancy of an HMM using probabilistic bisimulation. Bisimulation is a notion of equivalence between graphs, or as a relation between nodes of a single graph. The algorithm starts with a model with a large number of states and trains it using the Baum-Welch algorithm. After, the model is transformed to a labelled graph, the bisimulation algorithm is applied to the graph to reduce the number of states and a new model is obtained. Finally, the new model is retrained using the Baum-Welch algorithm. The algorithm's performance was compared to that of the Bayesian Information Criterion [Shw] which showed almost the same results in terms of classification accuracy, however, BIC is

more computationally demanding.

In the second method, sequential pruning [BMF03], the algorithm uses a model selection criterion such BIC or Minimum Description Length (MDL) [Rai86] and searches for a model that maximizes this criterion. First, the algorithm starts with a large number of states K_{max} and sets the minimum desired number of states for a model K_{min} . Iteratively, the algorithm estimates the current parameters using the Baum-Welch algorithm, computes its model selection value and prunes the least probable state. After $K_{max} - K_{min}$ iterations, the best model is the one that maximizes the model selection criterion. The main disadvantage of both methods is that they focus only on the number of states and consider a fixed topology and according to our experiments, the effect of number of states has less influence on the performance than the topology. Assuming equal computational cost for all previous algorithms, Stolcke and Brants algorithms would be the first choice, since they consider the model topology and the number of states.

2.1.4 Different approaches

A different approach for optimizing the structure of HMMs was based on measuring the distance between HMMs. Lyngso *et al.* [LPhN99] focused on comparing HMMs in terms of the co-emission probability of state emissions, Bahalman *et al.* [BBL01] used Bayesian estimates of HMM states correspondences and suggested the use of this measure as a criterion for selecting HMMs. Balasubramanian [Bal93] has done extensive work on finding equivalent HMMs based on equal probability of the observation sequence alone, and regardless of the number of internal states. He then used this result to define conditions and an algorithm for finding minimal generalized Markov model.

Another method that is based on model selection approach is the Discrimination

Information Criterion (DIC) [Bie03]. In this method, it is not desired to select the simplest model that best explains the data, but to select the model that is the less likely to have generated data belonging to competing classification categories [Bie03]. The method is considered discriminative in regard to the classification task because it uses the data that belong to competing classes and hence it introduces knowledge of the classification in the model selection process. Unlike Stolcke, Brants and Brand methods, this method focuses only on the number of states and the number of Gaussian mixtures per state, and it assumes a fixed topology for all models. Selecting HMMs based on the DIC increased the recognition results by 1.11% at the price of increasing the number of states and the number of parameters. The number of states increased by 14% and the number of parameters increased by 60% to achieve this performance. Obviously, the DIC is computationally demanding.

As will be shown in section 4.3, improving HMMs' performance using better parameter estimation algorithms received more attention than structure optimization methods. Also, some of the new training algorithms are getting more popular due to the improvement they provide. Their main advantage is that they require less computations than structure optimization methods and provide a reasonable improvement in terms of performance while keeping the structure simple. An obvious observation would be to combine structure optimization methods and new training algorithms to achieve better performance, however, to the best of our knowledge, we have not found this kind of work. In the following section, we show that improving time-series data classification using HMMs took a new approach. The new approach does not exclude HMMs but reduces their role to efficiently model the time-series data, and leave the task of classification to discriminative models such as SVMs and MLPs. The main challenge in this approach, is what would be the discriminative and fixed size feature vector extracted from HMMs that contain sufficient statistics to classify

time-series data. In other words, since MLPs and SVMs have very good generalization in classification tasks, and since no fixed size feature vectors can be extracted from time-series data, then we can use HMMs to model this type of data, then extract fixed size vectors from the HMMs.

2.2 Generative and Discriminative Models

A new approach that appeared recently in the machine learning community is the framework of generative and discriminative models. The first comparison between both models was introduced in [RH97]. Their analysis showed that learning discriminative models may not always lead to the best classifier. According to Bayes rule, generative models approximate the true density of the data likelihood, and with an absence of the class prior, the decision is taken based on the maximum likelihood. In case of the presence of the prior from the data, the classification becomes more accurate. For discriminative models, the focus on the posterior directly without considering the likelihood neither the prior, therefore, they ignore valuable information. They also proposed for the first time the new approach of combining both models for classification problems. However, the combination method was not clearly formalized and according to their paper [RH97]:

... this suggests a promising way of combining the two approaches: partition the feature space into two. Train a generative model on those dimensions for which it seems correct, and a discriminative model on the others.

A combination stimulated from [RH97] was introduced in [PMSM⁺03] where they used discriminative models (Ensemble of Neural Networks) to classify misclassified patterns from generative models (PCA).

A first formal combination appeared in [JH98]. They proposed a general method for extracting discriminatory features from generative models in general. The basic discriminative model in their work was the generalized linear model which is a general form of support vector machines. Generalized linear models require defining a kernel and hence was the need to derive the kernel function from generative probability model. Defining a kernel function implies assumptions about metric relations between patterns, therefore, these metric relations should be defined from the generative probability model. The basic idea was to capture the generative process in a metric between patterns by using the gradient space of the generative model. The gradient of the likelihood with respect to a parameter, or the Fisher score, describes how that parameter contributes to the process of generating a pattern, and it preserves all the structural assumptions that the model encodes about the generation process. An advantage of the Fisher kernel is its invariance to any invertible (and differentiable) transformation of the parameters.

The previous work focused on differentiating between the generating process of two different patterns and it did not include the class label in the process, however as mentioned in theorem (1) [JH98], if the Fisher kernel will be used as a kernel classifier, it will be asymptotically at least as good as a MAP based classifier. The Fisher kernel was recently applied to speech recognition and speaker verification in [QB02] where the probabilistic generative models were HMMs and the discriminative model was support vector machines.

Recently, thorough comparisons and rigorous analysis for the trade-off between generative and discriminative models were introduced in [NJ02] and [Bou03] respectively. In [NJ02], a comparison was carried out between generative and discriminative classifiers that studied the asymptotic error of each. It was shown that as the training set size increases, generative models indeed have a higher asymptotic error than the

discriminative classifier. Also, the generative model may also approach its asymptotic error much faster than the discriminative model. Faster in this context implies with less number of samples, possibly logarithmic in the number of parameters for the generative case. The final conclusion of the paper is that as the size of the training set increases, there are two regimes for performance, one in which each model does better. The first in which the generative classifier has already approached its asymptotic error and is thus doing better, and the second in which the discriminative model approaches its lower asymptotic error and does better.

In [Bou03], the comparison between both models is carried out at the level of parameter estimation and it proposes a new method for combining generative and discriminative models. The new method is called the generative-discriminative trade-off (GDT) estimate and it is based on a continuous class of cost functions that interpolate smoothly between the generative and the discriminative strategy. The preliminary results on real data showed that the intermediate model often gives much better classification performances than that of the discriminative and generative classifiers.

It can be seen from the above comparisons and investigations that combining both models will complement their individual powers. The proposed framework in this thesis is in general stimulated from [JH98] in that generative models are used to map the variable length sequential data into a single fixed size vector using the likelihood score instead of the Fisher score. Despite of the simpler combination method proposed, the preliminary experiments on the framework boosted the results of standard HMM by 3.83% which shows the potential of the generative-discriminative trend.

Chapter 3

Basics & Notations

This chapter introduces the basic theory of hidden Markov models, graphical models and support vector machines. Hidden Markov models form the core of this thesis so their basic theory comes in the first section. Section 2 reviews briefly the basics of graphical models in general and generative models in specific. Section 3 reviews the basic theory of support vector machines since they form the second stage of the proposed framework in this thesis. Finally, the notations that will be used throughout the thesis are illustrated in section 4.

3.1 Hidden Markov Models

Hidden Markov Models (HMMs) [BP66, Rab89, Ben99] are a class of stochastic processes that is capable of modelling time-series data. They belong to a larger class of models known as generative models. An HMM model λ can be seen as a single, discrete, hidden and multinomial variable that changes its state with time according to a certain distribution. At each state, the variable emits a symbol according to another distribution depending on the state of this hidden variable. The resulting model in that sense is a doubly embedded stochastic process with an underlying stochastic

process that is not directly observable but can only be observed through another set of stochastic processes that produce the sequence of observations [Rab89].

3.1.1 Elements of HMMs

A first order, time homogeneous HMM is defined by the following elements:

- A set of n random variables (states) $\mathbf{Q} = \{Q_1, \dots, Q_n\}$.
- An initial state probability distribution for $t = 1$ defined by:

$$\pi_i = Pr(q_1 = Q_i) \quad (1)$$

such that $1 \leq i \leq n$ and $\sum_{i=1}^n \pi_i = 1$.

- A state transition probability distribution defined by a transition probability matrix $A = \{a_{ij}\}$ that represents the probability of moving from state i to state j .

$$A = \{a_{ij}\} = Pr(q_t = Q_j | q_{t-1} = Q_i) \quad (2)$$

such that $1 \leq i, j \leq n$ and $\sum_{j=1}^n a_{ij} = 1$.

- An Observation probability distribution defined by $B = \{b_j(\cdot)\}$ such that:

$$b_j(z_t) = Pr(O = z_t | q_t = Q_j) \quad (3)$$

For discrete density HMMs, the observation $z_t \in V = \{v_1, \dots, v_M\}$, so that $z_t = v_m$. Therefore :

$$B_{v_m,j} = b_j(v_m) = Pr(O = v_m | q_t = Q_j) \quad (4)$$

such that $\sum_{m=1}^M b_j(v_m) = 1$.

For continuous density HMMs, the observation $z_t \in \mathbb{R}^D$ and the pdf of the state is usually represented as finite mixture form as follows:

$$B_{z_t,j} = b_j(z_t) = \sum_{k=1}^M C_{jk} \mathcal{N}(z_t, \mu_{jk}, U_{jk}) \quad (5)$$

where C_{jk} is the mixture coefficient for the k th mixture in state j , M is the number of mixtures in state j and \mathcal{N} is any log-concave or elliptically symmetric density such as the Gaussian density. Eq. 5 considers that the mixture Gaussian with a mean vector μ_{jk} and a covariance matrix U_{jk} for the k th mixture component in state j . The mixture weight (coefficient) C_{jk} satisfies the stochastic constraint $\sum_{k=1}^M C_{jk} = 1$ where $1 \leq j \leq n$, $1 \leq k \leq M$ and $0 \leq C_{jk} \leq 1$. For convenience, the complete set of HMM parameters are defined using the compact notation: $\lambda = (A, B, \pi)$.

It is worth noting that a discrete observation HMM can be obtained from a continuous one using vector quantization or clustering methods such as Learning Vector Quantization (LVQ) [Koh97] or K-Means clustering algorithm respectively.

For a classification problem with P classes, one usually builds an independent HMM for each class to maximize the likelihood of the training sequences labelled for that class, and then the classification is done using the Bayesian criterion. A new pattern Z_0 that belongs to class C_j is correctly classified if:

$$Pr(Z_0|\lambda_j) = \max_{p=1,\dots,P} Pr(Z_0|\lambda_p) \quad (6)$$

In real world applications, three main assumptions are considered when using HMMs for modelling time-series data. First, it is assumed that an event can cause another event in the future, but not vice-versa. This tends to simplify the design of the

HMM [Gha97]. Second, it is assumed that the underlying Markov chain is a time-homogeneous process, i.e. time-independent. A time-independent Markov process means that the state transition probability matrix is fixed throughout all the process and does not change according to another distribution as time-dependent Markov chain. This assumption makes capturing the dynamics of the model much easier, the number of parameters is much smaller, and the model can be trained on sequences of certain lengths and generalize to sequences of different lengths [Ben99]. Finally, it is always assumed that the HMM is a first order Markov chain. A higher order HMM becomes quickly intractable for large orders since the number of parameters required for an n state variable is $O(n^{k+1})$, where k is the order of the Markov chain. This restricts one to use a small value of k . However, most observed sequential data do not satisfy the Markov property for a small k , but as mentioned above, that the event can cause another event in the future and not vice-versa, warrant the hypothesis that at time t , past data in the sequence can be summarized concisely by the state variable, and this is the real embedded operation in HMMs. The observed data sequence is not assumed to have a low order Markov property, rather another hidden variable is assumed to exist and to have the Markov property of order 1. A first order HMM can emulate an HMM of any other order by increasing the number of values the hidden (latent) variable can take [Ben99], i.e. increasing the number of states.

3.1.2 Observation likelihood, optimal state sequence and training

In addition to the above formal definition of HMMs, there are three basic issues regarding the model in order to make it useful in real world applications. These issues are as follows:

Observation likelihood : Given a time-series pattern (observation sequence) Z and

a model λ , it is desired to answer the question of *How likely the observation Z was generated by the model λ ?*, i.e. compute the quantity $Pr(Z|\lambda)$. The forward and backward operators [Rab89] are standard recursive algorithms for computing this quantity, and they are used extensively in the Baum-Welch training.

Optimal state sequence : Given a time-series pattern Z and the model λ , it is desired to answer the question of *What is the optimal state sequence that best explains the observation ?*. The main problem in this question is the formal definition for the optimality criterion. The most widely used criterion is to maximize $Pr(Q, Z|\lambda)$. A formal technique for finding this single best state sequence is based on the dynamic programming method and it is known as the Viterbi algorithm [Rab89, Vit83].

Training : Training, parameter estimation or learning are three synonym terminologies for one problem, that is *How to adjust the parameters of the model λ to maximize the likelihood $Pr(Z|\lambda)$* . The first parameter estimation algorithm for HMMs and the most widely used is the Baum-Welch [BP70, Rab89] algorithm which is a form of the generalized Expectation-Maximization (EM) [DLR77] algorithm. The Baum-Welch and the EM algorithms are parameter estimation algorithms that maximize the likelihood of the data which do not guarantee the minimum classification error. This drawback of the algorithm encouraged researchers to develop new algorithms with different criteria than maximizing the likelihood of the data and that suites classification problems such as . Maximum Mutual Information [BBdSM86], Corrective Training [BBdSM88], Maximum A Posteriori [GL92], Entropy based Distance Functions [SW96], Minimum Classification Error (MCE) [JK92, KL98] and Factor Analysis & Minimum Classification Error [SR00]. Recently the MCE got the attention of a few researchers



Figure 1: A simple graphical model for two random variables.

and the algorithm is getting more popular [Bie02, GH02].

3.2 Graphical Models

Graphical models [Hec96] are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur in applied mathematics; uncertainty and complexity. Probability theory provides the glue that combines parts together and ensures the system’s consistency. On the other hand, the graph theory framework provides a way to view all these systems as instances of a common underlying formalism. This view has many advantages, in particular, specialized techniques that have been developed in one field can be transferred between research communities and exploited more widely. Fundamental to graphical models is the notion of modularity - a complex system is built by combining simpler parts.

Graphical models are graphs in which nodes represent random variables and the need of arcs represent conditional independence assumptions. Figure 1 shows a simple graphical model for 2 random variables X and Y . The arrow from X to Y can be interpreted as X causes Y , hence directed cycles are disallowed. We adopt the convention used in [Mur01] that squares nodes represent discrete random variables, and circles represent continuous random variables. Shaded nodes are observed variables, and clear ones are hidden.

The conditional independence assumption allows the joint probability distribution over the random variables to be represented in a factored form. Such a representation has two advantages 1) the model has fewer parameters to be estimated and consequently makes learning easier and faster, and 2) makes inference faster.

There are two main kinds of graphical models: undirected and directed. Undirected graphical models, also known as Markov Networks or Markov random fields, are used in physics and vision communities. Directed graphical models, also known as Generative Models or Bayesian/Belief Networks, are used in AI and Machine Learning communities. HMMs belong to this type of graphical models. The following subsection, discusses in brief generative models and gives several examples of different types of HMMs and their representation as directed graphical models.

3.2.1 Generative models

Generative models (GM) are directed graphical models for probabilistic relationships among a set of random variables. Such models record qualitative influences between variables in addition to the numerical parameters of the probability distribution. There are numerous representations for data analysis like rule bases, decision trees, and neural networks, also, there are many techniques for data analysis such as density estimation, classification, regression, and clustering; but generative models have some advantages over these representations and techniques. First, GM can handle incomplete data sets, i.e. when some variables are missing their values. This is due to their ability to encode correlation between input variables. Second, GM allows one to learn about causal relationships which is important to a) understand the problem domain, and b) allows to make predictions in the presence of interventions. Third, they facilitate the combination of domain knowledge and data. Formally, GMs are capable of learning and exploiting relations between a large set of variables due to

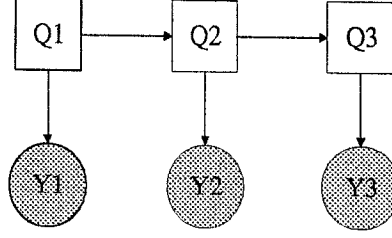


Figure 2: Discrete hidden Markov model unrolled for 3 time slices.

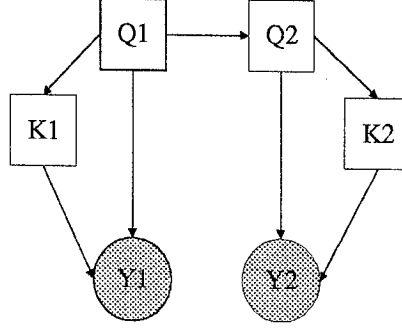


Figure 3: An HMM with mixture of Gaussians output unrolled for 2 time slices.

their ability to encode the joint probability distribution between these variables.

A GM for a set of variables $\mathbf{Q} = \{Q_1, \dots, Q_n\}$ consists of (1) a network structure S (directed acyclic graph) that encodes a set of conditional independence assertions about variables in \mathbf{Q} , and 2) a set Pr of local probability distribution for \mathbf{Q} . The nodes in S are in one-to-one correspondence with the variables in \mathbf{Q} . Pa_i is used to denote the parents of Q_i in S . The joint probability distribution for \mathbf{Q} is given by $Pr(\mathbf{Q}) = \prod_{i=1}^n Pr(Q_i | Pa_i)$.

Figures 2, 3, 4 and 5 show different types of HMMs from the literature such as discrete HMMs, continuous HMMs, IOHMMs [BF96] and Factorial HMMs [GJ97] with their representation as directed graphical models.

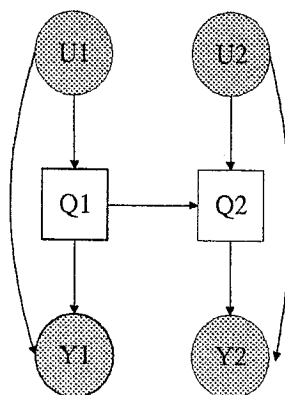


Figure 4: Input-Output HMM (IOHMM) unrolled for 2 time slices.

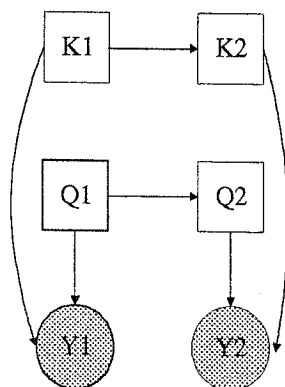


Figure 5: Factorial HMM unrolled for 2 time slices.

3.3 Support Vector Machines

Support vector machine is a new type of pattern classifier based on a novel statistical learning technique proposed by Vapnik [Vap98]. The appeal of SVMs is two-fold. Firstly, the process of tuning the parameters in the training algorithm is simpler, and secondly, they showed great ability in generalization performance not only in classification tasks but also in regression and density estimation problems. Due to their good generalization, SVMs have been successfully applied to a number of pattern recognition applications such as face detection, verification and recognition [NG02, WCH02], object recognition [SG01], character recognition [ZLX00, ACS02b], speech and speaker verification and recognition [WC00], information and image retrieval, gender classification [ZLZ01] and text categorization [Joa99b].

We recall that classical learning approaches are designed to minimize the error on the training dataset following by that what is called "*Empirical Risk Minimization*" and Neural Networks are the most common example on such an approach. SVMs, on the other hand, follow a different approach by minimizing the empirical risk and a term known as the VC confidence. The Empirical risk $R_{emp}(\alpha)$ and the VC confidence are known to be the "*Risk Bound*". SVMs minimize the Risk Bound via the Structural Risk Minimization approach, a piecewise minimization process for the Risk Bound.

For this section we follow the notation used in [Vap98]. Given a training set of l observations, and that each observation consists of a pair $\{X_i, Y_i\}$ $i = 1, \dots, l$ where $X_i \in \mathbb{R}^P$ and $Y \in \{-1, +1\}$. It is considered that there exists some unknown probability distribution $Pr(X, Y)$ from which these data are drawn and are assumed to be "iid". Suppose we have a learning machine whose task is to learn the mapping $X_i \rightarrow Y_i$. The machine is defined as a set of mappings $X \rightarrow f(X, \alpha)$ where $f(X, \alpha)$ themselves are labelled by the adjustable parameters α . A particular choice of α generates what is called a *Trained Machine*. Also we consider a 0-1 loss function

defined as: $\frac{1}{2}|Y_i - f(X_i, \alpha)|$. The expectation of the test error for this machine “*Expected Risk*” is defined by:

$$R(\alpha) = \int \frac{1}{2}|Y_i - f(X_i, \alpha)|dPr(X, Y) \quad (7)$$

The *Empirical Risk*, $R_{emp}(\alpha)$, is defined to be the measured mean error rate on the training set:

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^l |Y_i - f(X_i, \alpha)| \quad (8)$$

For a certain value η where $0 \leq \eta \leq 1$ and for the 0-1 loss function defined above, then with a probability of $1 - \eta$ the following bound holds [Vap98]

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(\frac{2l}{h}) + 1) - \log(\frac{\eta}{4})}{l}} \quad (9)$$

where h is a nonnegative integer called the Vapnik-Chervonenkis (VC) dimension, and is a measure of the capacity of the machine. The capacity of the machine is defined as the ability of the machine to learn a training set without errors. The right hand side of Eq. 9 is the Risk Bound and the the second term of the Risk Bound is the VC confidence.

3.3.1 SVM for linearly separable case

The basic idea of SVMs is to construct a hyperplane as the decision plane which separates the *+ve* examples and the *-ve* ones with the largest margin. In a binary classification problem, it is desired to find the separating hyperplane which gives the smallest generalization error among an infinite number of possible hyperplanes. Such a hyperplane is the one with the maximum margin of separation between the two classes, where the margin is the sum of distances from the hyperplane to the closest

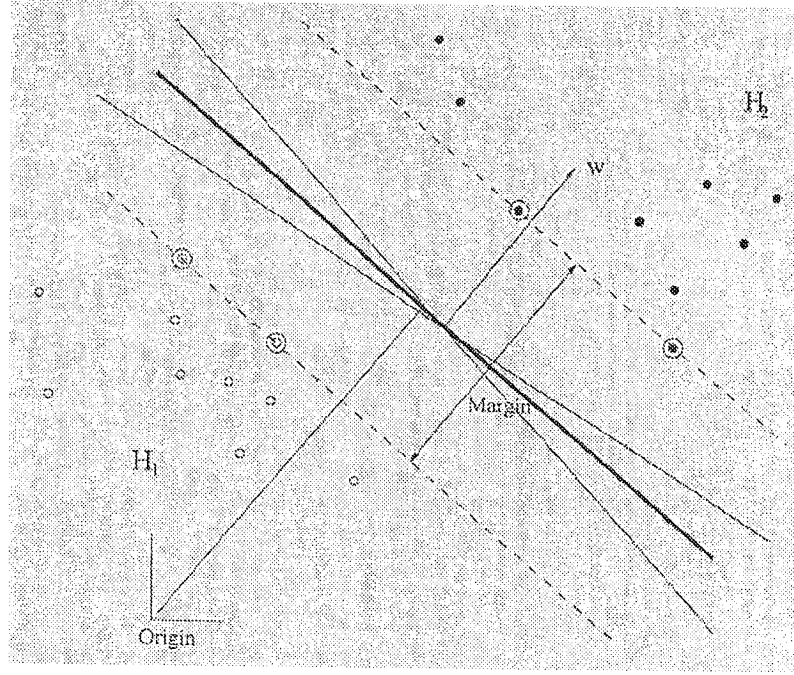


Figure 6: Among the separating hyperplanes, SVM chooses the one which has the largest margin. The support vectors are circled.

data points of each of the two classes. Figure 6 shows an example for such a case where the best hyperplane to classify the data is the one exactly in the middle (the thick line).

The data points that lie on the hyperplane are called the support vectors because they support the decision boundary between classes and they define the decision function of the SVM. Figure 7 shows the linear separable case for SVM. Suppose the data are completely separable by a P -dimensional hyperplane defined by:

$$W.X + b = 0 \quad (10)$$

where W is a vector normal to the hyperplane known as the weight vector, b is the bias term and $\frac{|b|}{\|W\|^2}$ is the perpendicular distance from the origin to the hyperplane. The separation problem is to determine the hyperplane such that $W.X_i + b \geq +1$ for $+ve$ examples and $W.X_i + b \leq -1$ for $-ve$ examples. Such a hyperplane can be found

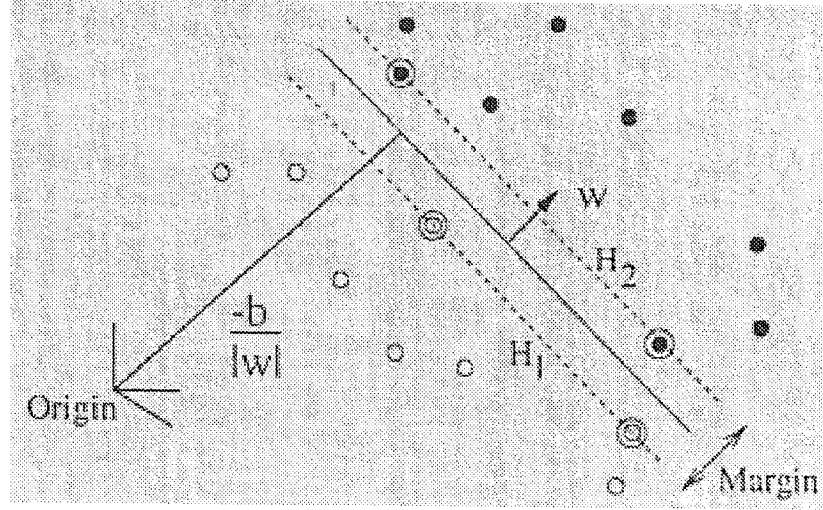


Figure 7: Linear separating hyperplanes for the separable case. The support vectors are circled

by maximizing $\frac{1}{\|W\|^2}$ which implies minimizing the following objective function:

$$\min_W L(W) = \frac{1}{2} \|W\|^2 \quad (11)$$

under the constraint:

$$Y_i(X_i \cdot W + b) - 1 \geq 0 \quad \forall i \quad (12)$$

This is a quadratic programming problem that can be solved using standard techniques such as Lagrange multipliers and Wolfe dual [CST00].

3.3.2 SVM for non-separable case

In real-life data, the two classes are not completely separable, but a hyperplane that maximizes the margin while minimizing a quantity proportional to the misclassification errors can still be determined. This can be done by introducing a *+ve* slack variable ξ_i in constraint 12, which then becomes:

$$Y_i(X_i \cdot W + b) \leq 1 - \xi_i, \quad \forall i \quad (13)$$

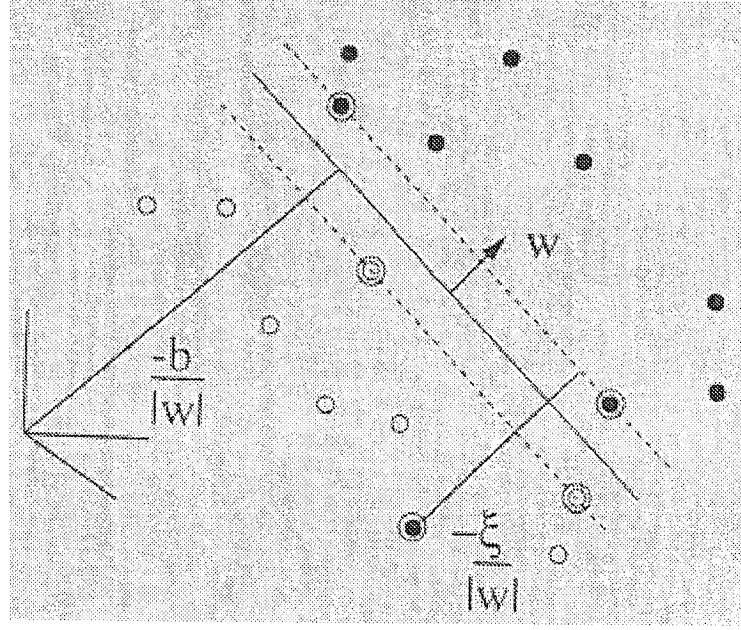


Figure 8: Linear separating hyperplanes for the nonseparable case

If an error occurs, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound for the number of misclassification errors. Hence the objective function $L(\cdot)$ becomes:

$$L(W, \xi) = \frac{1}{2} \|W\|^2 + C \sum_{i=1}^l \xi_i \quad \forall i \quad (14)$$

where C is a parameter chosen by the user that controls the trade-off between the margin and the misclassification errors. A larger C means that a higher penalty to misclassification errors is assigned. Figure 8 shows the nonlinear separable case for SVMs.

3.3.3 Nonlinear SVMs and kernels

In order to accomplish nonlinear decision function, an artificial mapping ϕ of the data into a very high (possibly infinite) dimensional Euclidean space \mathcal{H} is performed as $\phi: \mathbb{R}^P \rightarrow \mathcal{H}$, and the linear classification problem is performed in the new space with dimension D . Cover's theorem states that if the transformation is high enough, then

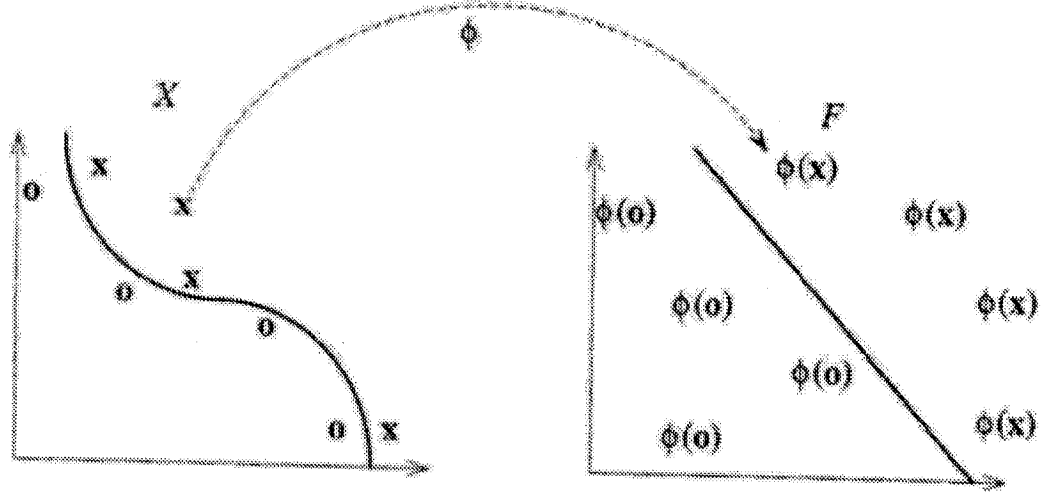


Figure 9: The role of the mapping function ϕ in making the data more linearly separable

the input space may be transformed into a new feature space where the patterns are linearly separable with high probability. Figure 9 shows the idea of mapping the data from the input-space to the feature-space where they become linearly separable. The training algorithm then only depends on the data through the dot product in \mathcal{H} of the form $\phi(X_i) \cdot \phi(X_j)$. Since the computation of the dot product is prohibitive in this new high dimension space, and since ϕ is unknown, Mercer's theorem for positive definite functions allows to replace $\phi(X_i) \cdot \phi(X_j)$ by a positive definite symmetric kernel function $K(X_i, X_j)$, that is, $K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$. In that sense, the data can become separable in the feature space although the original input is not linearly separable in the input space. Therefore, kernel substitution provides a route for obtaining nonlinear algorithm from algorithms previously restricted to handling linear separable data. Variant learning machines are constructed according to different kernel functions $K(X_i, X_j)$ and thus different hyperplanes in the feature space. Table 1 shows four different kernels and their inner product functions.

Table 1: Summary of inner-product kernels

Kernel Function	Inner Product Kernel
Polynomial Kernel	$K(X, X_i) = (X^T X_i + 1)^2$
Gaussian Kernel	$K(X, X_i) = \exp(-\ X - X_i\ ^2 / 2\sigma^2)$
Sigmoid Kernel	$K(X, X_i) = \tanh(\beta_0 X^T X_i + \beta_1),$ where β_0 and β_1 are user defined parameters
KMOD Kernel [ACS02a]	$K(X, X_i) = a[\exp(\frac{\gamma}{\ X - X_i\ ^2 + \sigma^2}) - 1]$ where a is a normalization constant equal to $\frac{1}{\exp(\gamma/\sigma^2) - 1}$

3.3.4 SVM applied to multiclass classification

The SVM is originally a binary classifier, however it should be extended to multiclass problems. There are two basic strategies for solving P -class problems with SVMs: 1) The one-against-all strategy, and 2) tree-structured (pairwise and DDAG) strategy.

One-against-all

Take the training samples with the same label as one class and the others as the other class, then it becomes a two class problem. For a P -class problem, P classifiers are formed and denoted by SVM_i , $i = 1, \dots, P$. For testing a sample X_0 , $d_i(X_0) = W_i \cdot X_0 + b$ can be obtained by SVM_i . The testing sample X_0 belongs to the class j where

$$d_j(X_0) = \max_{i=1, \dots, P} d_i(X_0) \quad (15)$$

Pairwise and DDAG

In the pairwise approach, $P(P-1)/2$ SVMs are trained to classify a testing example and the final result is synthesized from all these classifiers. The pairwise classifiers are arranged in trees, where each tree node represent a SVM. The Bottom-Up tree which is similar to the elimination tree used in tennis tournaments was originally proposed in [GZL01]. An alternative Top-Down tree called Decision Directed Acyclic Graph

(DDAG) has been recently proposed in [PCST00]. There is no theoretical analysis of the two strategies with respect to the classification performance. Regarding the training effort, the one-against-all strategy is preferable since it requires less number of classifiers to be trained.

3.4 Notations

In order to avoid any confusion for the reader, this section illustrates the basic notations that will be used in the rest of this thesis unless stated otherwise.

- A training set of time-series data is defined by $\mathbf{Z} = \{Z_i | 1 \leq i \leq N\}$ such that $Z_i = \{z_t^i | 1 \leq t \leq T_i\}$, $z_t^i \in \mathbb{R}^D$ and T_i is the length of sequence i .
- The pair $\{Z_i, Y_i\}$ represent the training example Z_i with the label Y_i such that $Y \in \mathcal{Y} = \{C_j | 1 \leq j \leq P\}$ where P is the number of classes.
- The set $\mathcal{Z}_j = \{Z_k | 1 \leq k \leq l_j\}$ such that $Z_k \in C_j$ and $l_j = |\mathcal{Z}_j|$.
- The function $\mathcal{F} : \mathbb{R}^D \rightarrow \mathbb{R}^P$ is a nonlinear function that takes Z_i as input and maps it to a point $X_i \in \mathbb{R}^P$. Therefore the set $\mathcal{X} = \{X_1, \dots, X_N\}$ is the image of the set \mathbf{Z} under the function \mathcal{F} .

It is worth noting that it is not a restriction that the dimensionality of X should be equal the number of classes. It could be that due to several variations in patterns within one class, the data could be represented using more than one model. In that case, the number of models is going to be larger than the number of classes and X will have a dimensionality equal to the number of models. However, here we used the simple case where each class is represented by one model.

Chapter 4

Notes on Hidden Markov Modelling

This chapter provides a deeper insight into hidden Markov modelling by focusing on the representation capability of HMMs. The main issue covered in this chapter is understanding the factors that affect the modelling capability of HMMs and how it can affect the final performance of HMM-based classifiers.

The modelling capability of HMMs is affected by two main factors: (I) the model structure, and (II) the parameter estimation or the learning (training) algorithm. The model structure can also be split to two factors: (a) the number of states, and (b) the topology, i.e. the connections between the states. In addition to the mentioned factors, there are other factors that arise from the nature of the model itself and they have a direct impact on the modelling capability. Sections (1), (2) and (3) discuss all the above mentioned issues in more details.

4.1 Hidden Markov Modelling Limitations

We discuss in the following some limitations that arise from the nature of hidden Markov modelling. The first limitation of HMMs is that they represent the recent history of the sequence using a single, discrete n -state multinomial variable. The efficiency of the Baum-Welch re-estimation depends on this fact, but it severely limits the representational power of the model. The hidden state of a single HMM can only convey $\log_2 n$ bits of information about the recent history. If the HMM had a distributed hidden state representation [BH01] consisting of m variables each with n alternative states it could convey $m \log_2 n$ bits of information. So, the information bottleneck scales linearly with the number of variables and only logarithmically with the number of states of each variable [BH01]. This simply suggests that in a multi-class problem, it is better to decompose the class to subclasses and use many small HMMs to model each subclass (mixtures of models), where the outputs are somehow combined to produce a single output for the class.

Second, despite that HMMs have a very powerful relationship between the underlying state and the associated observations because each state stores a private distribution over the output variables, any change in the hidden state can cause complex changes in the output distribution. Consequently this makes it extremely difficult to capture reasonable dynamics on the discrete hidden (latent) variable because in principle any state is reachable from any other state at any time step and the next state depends on the current one [Row99].

4.2 The Effect of the Structure

In the following, we investigate the effect of the topology on the modelling capability of HMMs. We studied the effect of the topology from two different perspectives: (1) the

graphical models framework, and (2) the diffusion of credits while learning Markovian models. The experimental results supporting the theoretical results are illustrated in Chapter 6 (Experimental Results). Investigating the effect of number of states on the modelling capability of HMMs is based on extensive empirical experiments only, therefore they are also illustrated in detail in Chapter 6

4.2.1 The topology factor

In many applications that use HMMs, the number of states is manually predetermined prior to training. The connections between states, (topology) are determined by setting non-zero probabilities in the A matrix prior training. During training, the Baum-Welch (EM) [BP70] algorithm improves the estimates of these probabilities from the data. Note that the EM algorithm can not set 0 or 1 (can approach 0 or 1) probabilities in the A matrix, therefore it can not be seen as an algorithm that learns the topology.

An approach that can determine automatically from the data the number of states of the model and the interconnections between them, i.e. learning the number of states and the topology, is the model selection approach. Using this approach the problem can be formulated as follows. Given the training set of examples \mathcal{Z} and a criterion function Υ for the quality of the model on the data set \mathcal{Z} , choose a model from a certain set of models, in such a way to maximize the expected value of this criterion function on new data (assumed to be sampled from the same unknown distribution from which the training data was sampled) [Ben99].

Bayesian formulation for model selection

HMMs can be viewed as a special case of Graphical Models [Hec96, Mur01]. Model selection is one of the main problems in graphical models and much work has been introduced regarding this problem. The Bayesian approach, one of the main approaches for model selection, is a fundamental approach for model selection in graphical models. Following this approach means encoding the uncertainty about the structure of the HMM by using a discrete variable whose states correspond to the possible HMM-structure hypotheses S^h and assessing it the a priori density $P(S^h)$. Given the training example set \mathcal{Z} for the model λ and augmenting the model parameters A, B, π in a single parameter vector θ , the problem would be computing the posterior distribution for the HMM structures. This can be formulated as follows using Bayes theorem:

$$Pr(S^h|\mathcal{Z}) = \frac{Pr(\mathcal{Z}|S^h)Pr(S^h)}{Pr(\mathcal{Z})} \quad (16)$$

where $Pr(\mathcal{Z})$ is a constant that does not depend on the network structure. The maximum likelihood structure would be the complete graph [Mur01], i.e. the full ergodic model, since this has the greatest number of parameters, and hence can achieve the highest likelihood. On the other hand this increases the model's complexity and will let the model overfit the training data resulting in a poor generalization. In fact, the marginal likelihood in Eq. 16 plays an important role to prevent this overfit. From the definition of the marginal likelihood:

$$Pr(\mathcal{Z}|S^h) = \int Pr(\mathcal{Z}|S^h, \theta)Pr(\theta|S^h) d\theta \quad (17)$$

it automatically penalizes more complex structures since they have more parameters and hence cannot give as much probability mass to the region of space where the data actually lies. In other words, a complex model is less believable and hence less likely

to be selected. This phenomenon is known as Ockham's Razor [Mur01] which favours simple models over complex ones. It can be seen that though the number of states may be fixed, the topology can affect the modelling capability in a serious way.

Diffusion of credits in Markovian models

A different approach that studied the effect of the topology on the modelling capability of HMMs was presented in [BF95]. Bengio and Frasconi investigated the problem of diffusion in homogeneous and non-homogeneous HMMs and its effect on learning long term dependencies. Training HMMs requires propagating forward and backward probabilities and taking products of the transition matrix A . Therefore, two types of diffusion exist, diffusion of influence in the forward path and diffusion of credit in the backward phase of training. Their paper studied under which conditions these products of matrices will converge to a lower rank, thus harming learning long term dependencies. The difficulty of learning was measured by using the Dobrushin's ergodicity coefficient [Sen86] defined as follows:

$$\tau(A) = \frac{1}{2} \sup_{i,j} \sum_k |a_{ik} - a_{jk}| \quad (18)$$

where $A = \{a_{ij}\}$ is the transition probability matrix. It was shown that in all cases, while training HMMs, the ergodicity coefficient will converge to 0 indicating a greater difficulty in learning, but the rate of convergence depends on the topology. Figure 10 [BF95] shows the convergence of 4 HMMs with the same number of states but with different topologies. It can be seen that the full ergodic model has the fastest convergence rate and that simpler models are slower. The final conclusion is that in order to avoid any kind of diffusion, most transition probabilities should be deterministic (0 or 1 probability). The result coincides with the Ockham's Razor result obtained from the previous section and both prefer simple topologies to complicated ones.

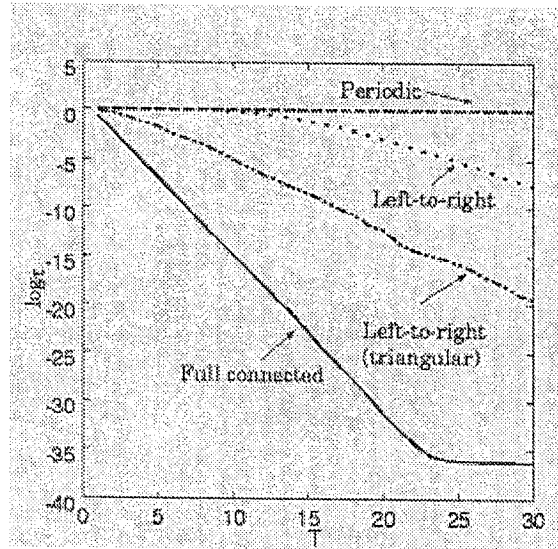


Figure 10: Convergence of Dobrushin's coefficient for 4 different topologies.

4.3 Parameter Estimation

This section discusses the effect of parameter estimation algorithms on the performance of HMMs when used for classification. Prior to the discussion, a review on some of the well known parameter estimation algorithms for HMMs is introduced.

Recall that for a P -class problem, each class is represented by a single model (assuming simple case) independent from other models. Also, consider that the parameters A, B, π of the model of λ_j for class C_j are stacked in single parameter vector $\theta_j \in \Theta$ where Θ is the parameter space. Using this notation, we review the different training algorithms for HMMs in a chronological order:

Baum-Welch (ML) [BP70]: The Baum-Welch algorithm was the first training algorithm for HMMs. It is based on the Maximum Likelihood (ML) approach that is achieved by means of the Expectation-Maximization algorithm (EM) [DLR77]. The algorithm considers that θ_j is fixed but unknown and iteratively,

the algorithm finds θ_j that maximizes the likelihood of the data as follows:

$$\hat{\theta}_{ML} = \arg \max_{\theta_j} Pr(\mathcal{Z}_j | \theta_j) \quad (19)$$

The Baum-Welch algorithm is guaranteed to monotonically increase the likelihood function and many of the following algorithms depend on it as a first step in the training.

Maximum Mutual Information (MMI) [BBdSM86]: An initial estimation of the parameters of each model is done using the Baum-Welch algorithm. Then, the MMI algorithm changes the initial estimates of the parameters by maximizing the mutual information defined as:

$$I(\mathcal{Z}_j, \theta_j) = \log Pr(\mathcal{Z}_j | \theta_j) - \log \left(\sum_{i \neq j} Pr(\mathcal{Z}_j | \theta_i) Pr(\theta_i) \right) \quad (20)$$

Therefore:

$$\hat{\theta}_{MMI} = \arg \max_{\theta_j} I(\mathcal{Z}_j, \theta_j) \quad (21)$$

The MMI in that sense is better than the pure EM since its second step updates the initial estimates of the EM based on information from other models. A complete derivation of the new estimates of θ_{new} can be found in [BBdSM86]. However, MMI has a drawback as parameter estimation algorithm; unlike the EM which provably converges to a local maximum of the likelihood, the MMI has a lack of such a proof for its recursive estimators.

Corrective Training [BBdSM88]: The Corrective training can be described in the following steps:

1. Using the training data \mathcal{Z} of each model λ , use the Baum-Welch algorithm to obtain an initial estimate for the parameters of each model.

2. Perform a recognition for the whole training set using the models obtained from the previous step.
3. If any pattern $Z_i \in C_t$ was recognized as C_f , adjust the correct model of Z_i so as to make λ_t more probable and λ_f less probable, then perform step 2. Else stop the correction.

The algorithm is motivated from the error correction procedure for linear classifiers, but unlike the existence of a proof of convergence for linear classifiers, the authors were unable to derive such a proof, however they proved empirically that the algorithm is working well.

Segmental K-Means (SKM) [JR79]: The objective of the Segmental K-Means is to maximize the state sequence of the observation, i.e. $\max_Q Pr(Z_j, Q|\theta_j)$. Hence:

$$\hat{\theta}_j = \arg \max_{\theta_j} \{ \max_Q Pr(Z_j, Q|\theta_j) \} \quad (22)$$

Eq. 22 which is known as the state optimized likelihood, focuses mainly on the most likely state sequence as opposed to summing over all possible state sequences as in the ML approach. This tends to reduce the computational burden of the algorithm. Also, Eq. 22 can be seen as maximum a posterior sequential estimate. The algorithm was motivated from speech recognition problems and it was shown to be better than the Baum-Welch, however, to the best of our knowledge, it was only applied in speech recognition applications.

Maximum A Posteriori (MAP) [GL92]: The MAP estimate can be seen as a Bayesian estimate of the vector parameter when the loss function is not specified. As mentioned earlier, the ML approach considers θ_j to be fixed but unknown, however, the MAP approach assumes that θ_j is a random vector taking values from Θ . θ_j has to be estimated from \mathcal{Z}_j which has a p.d.f $f(.|\theta_j)$ and a

prior p.d.f $g(\cdot)$ for θ_j . Hence, $\hat{\theta}_{map}$ is defined as:

$$\hat{\theta}_{map} = \arg \max_{\theta_j} f(\mathcal{Z}_j | \theta_j) g(\theta_j) \quad (23)$$

The MAP could be estimated for mixtures of Gaussians and then it was deployed to continuous density HMMs which resulted in two algorithms; 1) The Forward-Backward MAP estimate, and 2) The Segmental MAP estimate.

Entropy Based Distance Function [SW96]: Instead of maximizing $Pr(\mathcal{Z}_j | \theta_j)$, the Entropy based Distance Function, iteratively, maximizes the following function:

$$U(\tilde{\theta}_j) = \eta \log(Pr(\mathcal{Z}_j | \theta_j)) - d(\tilde{\theta}_j, \theta_j) \quad (24)$$

where η is the learning rate, and $d(\tilde{\theta}_j, \theta_j)$ is the distance between the old and new parameters. The distance that was chosen in the algorithm was the relative entropy between the two parameter vectors. The motivation for Eq. 24 is as follows. Consider that in a batch training mode, and after several iterations, we have a parameter vector θ . In the next iteration, it is desired to keep the new parameter vector $\tilde{\theta}$ close to θ which incorporates all the knowledge obtained in past iterations, but it should also maximize the log-likelihood of the data set. Despite the theoretical proof of convergence that backs up the algorithm, and the promising results it showed, it was not popularized in the speech recognition community. This could be due to the appearance of the algorithm appeared in different communities and its high computational cost.

Minimum Classification Error [Bie02]: An initial parameter estimate for the model is done by using the Baum-Welch algorithm. For each model λ_i , we

define a discriminant function g_i as follows:

$$g_i(Z, \lambda_i) = [\log \Pr(Z|\lambda_i)]^{\frac{1}{v}} \quad (25)$$

where v is a positive constant. By choosing a large v , the discriminant function term is dominated by the lexeme of maximum score. By varying v , the contribution of each lexeme's score to the character discrimination function can be controlled. The recognition is done by choosing the class that has the highest discriminant measure:

$$\text{choose } C_i \text{ if } i = \arg \max_j g_j(Z, \lambda_j) \quad (26)$$

From the discriminant function, a misclassification measure can be derived:

$$d_i(Z, \Lambda) = \frac{-g_i(Z, \lambda_i) + \log[\frac{1}{P-1} \sum_{j, j \neq i} \exp g_j(Z, \lambda_j) \eta]^{1/\eta}}{T} \quad (27)$$

where η is a penalty term, P is the number of classes and hence the number of models and T is the sequence length of Z . Choosing a large $+ve\eta$ implies focusing on learning the most competing categories. As $\eta \rightarrow +\infty$, the misclassification measure is reduced to a difference in score between the best but incorrect category and the true category, i.e. learning becomes a two class classification in each iteration. From the misclassification measure, the MCE loss assigned to Z can be defined as:

$$\ell(Z, \Lambda) = f(d_i(Z, \lambda)) \quad (28)$$

where $f(\cdot)$ is a smooth approximation of the step-wise 0-1 loss function, which is equal to one for positive values and zero otherwise. The choice of $f(\cdot)$ should

reflect the variations of the error rate and should be smooth enough to allow minimization by standard gradient techniques. In [Bie02] the truncated sigmoid was chosen:

$$f(\tau) = \begin{cases} \frac{1}{1+\exp -\alpha\tau} & \text{if } \tau > \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

where α is *+ve* and ϵ is a threshold that controls the degree of discrimination during the MCE training. After defining the MCE loss function, the goal of the MCE training is to minimize the expected loss $L(\Lambda)$ of the overall parameter set defined over the training set:

$$L(\Lambda) = \frac{1}{N} \sum_{j=1}^P \ell(Z_{\in C_j}, \Lambda) \quad (30)$$

The Minimization of $L(\cdot)$ is usually done by the Gradient Probabilistic Descent method (GPD) which updates the parameters of the system using a token by token basis. A drawback of the GPD is its sensitivity to the tuning of the parameters such as the learning rate. An alternative choice could be a second order gradient method such as the Newton algorithm which is less sensitive to the learning parameter.

4.3.1 Discussion

Two remarks can be drawn from the above algorithms. First, the MMI, The Corrective Training and the MCE, share a common step, that is an initial estimate of the model parameters using the Baum-Welch algorithm. Secondly, all of the training algorithms train each model independently without incorporating any knowledge about other models. The two algorithms that overcome this problem are the MMI and the MCE since they modify the parameters according to some knowledge from other classes. Since the MMI has no proof of convergence, the MCE seems to be more

attractive. In the literature of HMMs, the training procedure is called *Parameter Estimation* which best describes the function of these algorithms despite their different criteria. This common feature is the main characteristic of generative models, that is, the training algorithm estimates the parameters of a predefined model by maximizing the likelihood (or any other criterion) of the data ($Pr(Z|\theta)$) independently from other models. Such a property has a strong effect on the performance of HMMs when used for classification, that is maximizing the likelihood or any other criterion function, does not guarantee less classification errors. For that reason, the MCE algorithm, lately has received more attention from researchers [SR00, GH02, Bie02].

On the other hand, it is well known from practical problems and the theoretical foundation, that for classification problems, it is better to estimate the posterior probability directly or learn a direct mapping from the data to their class labels. This is the characteristic of discriminative models that construct decision boundaries between classes. Unfortunately, discriminative models can not be used to classify time-series data due to the variable length of these sequences. The next chapter introduces a thorough comparison between generative and discriminative models in general, followed by the proposed framework for classifying time-series data.

Chapter 5

The Generative-Discriminative Framework

As discussed in the previous chapter, HMMs have a great ability to model time-series data, however, for classification problems, they have a degraded performance due to factors such as the structure and the type of the training algorithms. Also, it is very difficult to use SVMs and MLPs to represent this type of data. The difference in modelling capability and classification performance among HMMs, MLPs and SVMs is due to the type of each model. HMMs belong to the class of generative models that their primary use is in data modelling. SVMs and MLPs belong to the class of discriminative models that it's primary use is in discrimination (classification) problems.

This chapter proposes a new framework that combines the complimentary features of HMMs and SVMs for classifying time-series data. Section (1) recalls Bayes decision theory, and Section (2) presents a thorough comparison between generative and discriminative models in general. Finally, section (3) introduces the new framework.

5.1 Bayes Decision Theory

We review in brief the Bayes decision theory for classification. The classification problem consists of assigning a vector $X \in \mathbb{R}^P$ to one of the K classes. The true class is denoted by $Y \in \{C_1, \dots, C_K\}$. The classifier can be seen as a mapping between the observations and the labels. For each class, there is a true joint density $Pr(X, Y) = Pr(Y|X)Pr(X) = Pr(X|Y)Pr(Y)$ which is unknown in practice and the only available information is the training set $\{X_i, Y_i\}$ for $1 \leq i \leq N$ [RH97]. There is also a cost matrix $C(r, s)$ where $r, s = 1, \dots, P$ which describes the cost associated with misclassifying a member of class r to class s . A special case is the 0 – 1 loss, where $C(r, s) = 1$ if $r \neq s$ and 0 otherwise. The goal of the classification is to minimize the cost of errors, known as the *The Overall Risk* and in the case of a 0 – 1 loss, this can be achieved by assigning the observation X to the class with the maximum posterior probability:

$$C_j = \arg \max_{1 \leq j \leq P} Pr(Y = C_j|X) \quad (31)$$

where $Pr(Y = C_j|X)$ can be defined using the Bayes rule as:

$$Pr(Y = C_j|X) = \frac{Pr(X|Y = C_j)Pr(C_j)}{Pr(X)} \quad (32)$$

We also define the discriminant function

$$\gamma_j(X) = \log \frac{Pr(Y = C_j|X)}{Pr(Y = C_{i \neq j}|X)} \quad (33)$$

5.2 Generative Vs Discriminative Models

Choosing between a discriminative or a generative model is problem dependent. For a density estimation problem, generative models would be the best choice. However, for classification problems, discriminative models would be the first choice. A main reason for this choice that is succinctly articulated [NJ02] by Vapnik [Vap98], is that “*one should solve the classification problem directly and never solve a more general problem as an intermediate step such as modelling $Pr(X|Y)$* ”. Also in practice, discriminative models are preferred to generative ones due to their low asymptotic error. Despite of the low error rate in many classification problems that used discriminative models, it was shown in [RH97] that learning discriminative models might not always lead to the best classifier. In addition, it is very difficult to classify time-series data using discriminative models due to their variable length.

Since the paper is concerned with variable length time-series data but the goal is the classification of this type of data, it is better to have an insight of the advantages and disadvantages of generative and discriminative models from the following perspectives [QB02]:

Target of learning and the decision rule : Generative models learn a model of the joint probability density $Pr(X, Y)$, of the input X and the label Y and the prior density $Pr(Y)$. Their prediction is made by computing the likelihood $Pr(X|Y)$ using Bayes rule and then picking the most likely Y . On the other hand, discriminative classifiers focus on modelling the decision boundaries between classes by modelling the posterior probability $Pr(Y|X)$ directly or learning the direct map from input X to the class labels. Therefore, the focus of discriminative models is on correct classification only while generative models focus on modelling the true density of the data.

Learning method : For generative models, a model is chosen for the class densities $Pr(X|Y = C_j) = \mathcal{N}(X, \mu_j, \sigma_j)$ and the model parameters are estimated from the data by maximizing the full log likelihood

$$\theta_j^{MLE} = \underset{\theta}{argmax} \sum_{i=1}^{N_j} \log Pr(X_{\in C_j}^i, Y_i) \quad (34)$$

where N_j is the number of patterns in class C_j , μ_j and σ_j are augmented in the parameter vector θ_j . The EM algorithm is widely used algorithm for maximizing Eq. 34 and it has been proved it converges monotonically to a local maximum likelihood solution.

The discriminative approach is more flexible within regards to the class densities that it is capable of modelling. By observing Eq. 33 the generative model in terms of class densities can be seen as a special instance of the more general discriminative model. Parameter estimation in the discriminative case is carried out by maximizing the conditional log likelihood

$$\theta_j^{DISCR} = \underset{\theta}{argmax} \sum_{i=1}^{N_j} \log Pr(Y_i | X_{\in C_j}^i) \quad (35)$$

which focuses directly on the class posteriors. However, using this approach, a part of the data is ignored, namely, the marginal distribution $Pr(X)$. Compare the full likelihood $Pr(X, Y) = Pr(Y|X)Pr(X)$ with the conditional likelihood which ignores the second term of the right hand side. Therefore, if the class density is correct, the discriminative approach ignores important information. However, ignoring the class models may be good if they are not correct. The above observations are summarized in the following Table 5.2 [RH97]

Modular learning : For generative models, an independent model is built for each class where each model is trained individually and considers only the data whose

Table 2: Comparing Learning in Generative and Discriminative models

	<i>Generative</i>	<i>Discriminative</i>
<i>Objective Function</i>	Full log likelihood $\sum_i \log Pr(X_i, Y_i)$	Conditional log likelihood $\sum_i \log Pr(Y_i X_i)$
<i>Model Assumptions</i>	Class densities	Discriminant function
<i>Parameter Estimation</i>	Easy	Hard
<i>Advantages</i>	More efficient if model is correct	More flexible, robust because fewer assumptions
<i>Disadvantages</i>	Bias if model is incorrect	May also be biased Ignores information in $Pr(X)$

labels correspond to it. Hence, the model does not interact with other classes and avoids considering the whole training set and consequently learning is simplified and the algorithm proceeds faster. Moreover, the addition of a new class or deletion of an existing one is easier. Unlike generative models, discriminative models build a single model for all classes and hence it requires simultaneous consideration of all other classes which makes training harder, involve iterative algorithms and do not scale well [RH97].

Sample complexity : In the case of many models, such as the case of generative models in general, the VC dimension is roughly linear or at most low-order polynomial in the number of parameters [NJ02], however, it was shown in [Vap98] that the sample complexity for discriminative models is linear in the VC dimension.

Missing data : Unlike discriminative models, generative models are capable of learning even in the presence of some missing values. This is due to their learning method which optimizes the model over the whole dimensionality and thus models all the relationships between the variables in a more equal manner.

Rejection of poor or corrupted data : The likelihood value obtained from generative models is more reliable than the posterior obtained from discriminative

models, since generative models try to represent the true density of the data. A corrupted input or an outlier can be easily detected by the low likelihood and hence the design of a rejection rule is made easier.

5.3 The Proposed Framework

The above mentioned advantages and disadvantages led us to propose a new framework that combines the advantages of both models and overcomes the disadvantages of each separately. The framework consists of two stages, namely 1) the modelling stage, and 2) the classification stage. Figure 11 shows a block diagram of the proposed framework.

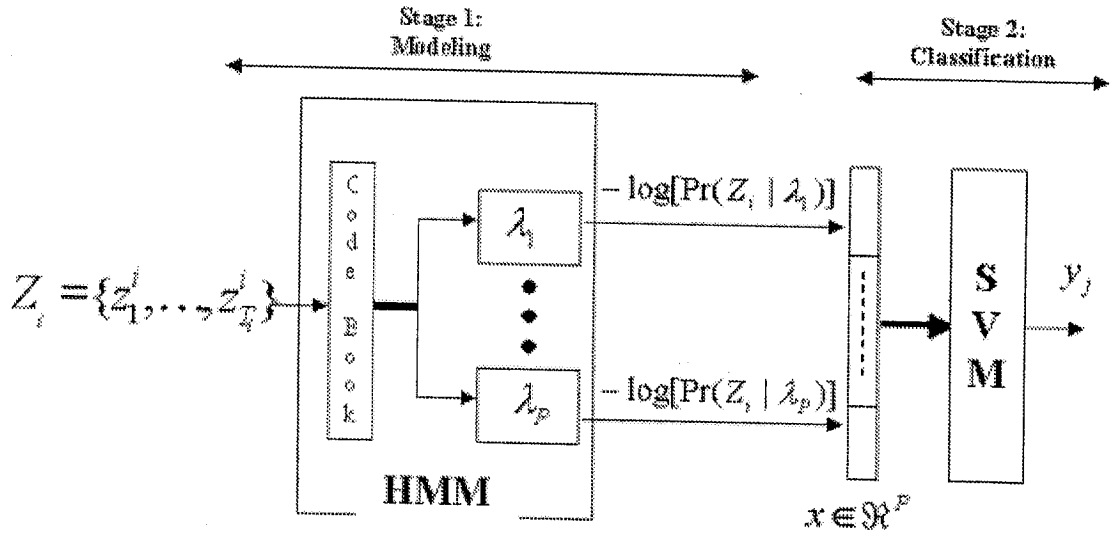


Figure 11: A block diagram of the proposed framework

The *modelling stage* is the first stage of the proposed framework and it consists of generative models. It has the basic role of mapping the variable length sequential pattern Z_i into a single fixed size vector $X_i \in \mathbb{R}^P$. The basic idea for the modelling stage is as follows. For a P -class problem, each HMM is trained with a set of examples that belong to its class. However, when using the classification rule of Eq. 6 to classify

a new input pattern Z_0 , each model λ_j is given the input pattern Z_0 to compute the forward probability $Pr_j(Z_0|\lambda_j)$ [Rab89] and the hope is always that the model of the correct class will output the highest likelihood. In the proposed framework, the modelling stage gets more information from all the models of the modelling stage in a P -dimensional real vector X (the likelihood score). In that sense, the modelling stage represents each sequential input as a point in the new space \mathbb{R}^P , or more formally, it can be considered as a nonlinear function \mathcal{F} such that $\mathcal{F} : \mathbb{R}^D \rightarrow \mathbb{R}^P$. Therefore the set \mathcal{X} is the image of the set \mathcal{Z} under the function \mathcal{F} .

The classification stage is the second stage of the proposed framework. It consists of a discriminative model that has the role of classifying the likelihood scores, the set \mathcal{X} , representing the sequential patterns. The discriminative model could be an MLP neural network, an SVM or any other discriminative model, however, we chose SVMs for their generalization capability. In fact, the discriminative stage acts as an ordinary classifier and its input is the output of the modelling stage which acts as a feature extraction layer. Increasing the discrimination between the generative models implies more discriminative feature vectors and consequently more accurate classification. Therefore, the modelling stage and the likelihood value are the key players of the framework. In the following, an insight of the likelihood value and the intuition lying behind the proposed model are elaborated in greater details.

5.3.1 The likelihood score

Consider the P -class problem in hand, each class is represented by a single HMM, and that the data (training set and test set) are i.i.d drawn from the same unknown distribution and they exist in an Euclidean space \mathcal{S} .

The set of the P models estimated from the training data form a set of local densities that allocate a certain part of the huge space \mathcal{S} . Although, it is desired

to have these densities far apart from each other in order to reduce the Bayes error, real life data (probably with noise and outliers) do not produce perfectly separated densities and ambiguities can exist easily.

The likelihood score of the HMM measures the closeness of the pattern to the model itself, or how likely the model generated this sequence. Consider the two classes C_j and C_i and the two sequences $Z_j \in C_j$ and $Z_i \in C_i$. For correctly trained models λ_j and λ_i , it should be that $Pr(Z_i|\lambda_j) < Pr(Z_j|\lambda_j)$ and the same for all other sequences that do not belong to C_j . It was claimed in [QB02] that HMMs can assign the same likelihood to two totally different sequences, however in practice, likelihoods can be close to each other and not equal and this closeness depends on the similarity between the two sequences and the sequences used for training this model. Consider for example the two sequences generated from handwritten digits 1 and 7 as in Figure 12. Therefore, the likelihood scores stored in X should have a high likelihood of the correct class and low likelihoods of other classes where each low value represents how the sequence is close to its model.

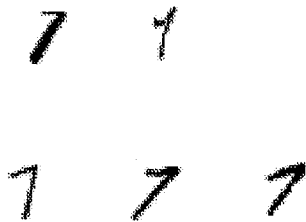


Figure 12: Handwritten digits: samples of digits 1 are shown in the first row, while samples for digit 7 are shown in the second row

The proposed approach is stimulated from this observation. For a pattern Z , each model votes for this pattern and instead of considering the highest likelihood as in Eq. 6, all the likelihoods are considered and taken as an input for a classifier that learns the voting of these models. Apparently, the method can be considered as a classifier combination scheme.

Chapter 6

Experimental Results

Recognition of unconstrained handwritten digits is an old and yet a well known problem in pattern recognition. Due to the extensive research done in this area, state-of-the-art techniques [LNSF03, SSP03] were able to achieve very low error rates on well known databases such as the MNIST database [LeC]. However, the problem is considered as a standard for testing new classifiers, learning algorithms and feature sets.

The MNIST database [LeC] is a very well known database for unconstrained handwritten digits that has a high variability in handwriting styles. The dataset has a training set of 60,000 samples and a test set of 10,000 samples from approximately 250 writers. The digits are stored in gray level images (0 – 255) that are cropped and scaled to be contained in a 20x20 pixels images. Figure 13 shows some samples from the training set of the MNIST database.

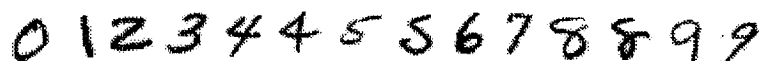


Figure 13: Samples from the training set of the MNIST data base.

This chapter illustrates all the experiments conducted in order to validate the investigation of this thesis. Section (1) introduces the features extracted from the images and how they were represented as time-series data. Section (2) shows the experiments that investigated the effect of the structure on the performance of HMM-based classifiers. The first part provides experimental results that support the theoretical results obtained in Section (4.2.1). The second part provides the experiments that investigated the effect of the number of states on the performance of HMM-based classifiers. Finally, Section (3) shows the results obtained from the proposed framework when used for recognizing unconstrained handwritten digits from the MNIST database.

6.1 Feature Extraction

The features extracted from the images should be represented in a time-series format. Therefore the well known sliding window technique [Cor96] was used for this purpose. First, the gray level values of the images were normalized to the range from 0 to 1. Second for each image a sliding window with a width of 3 pixels, height equals the image height was passed over each image from left to right. Each two successive windows were overlapped by two pixels. This resulted in an observation sequence length of 18 vectors from each image, i.e. that the 2-D image was transformed to a time-series pattern of continuous data where the length of the pattern is 18 observation vectors, and each observation has 20 variables. From each window (observation vector), a feature vector is extracted by computing the average gray level value in each row of the window, i.e. the sum of gray level pixels in each row divided by the window width and hence each vector consists of 20 features. All the experiments described in the following sections, were conducted on the time-series data obtained from the MNIST database with the method mentioned above.

6.2 The Effect of the Structure on the Performance

The experiments described in this section studied the effect of the structure on the performance of HMM-based classifiers. Two types of experiments were carried out, one to study the effect of number of states on the performance, and the other to study the effect of the topology on the performance.

6.2.1 HMM density type, initialization and codebook size

The experiments were conducted using discrete HMM (DHMM) based classifiers where each consisted of 10 DHMMs. The number of states for each model was determined according to the goal of the experiment. The topology used in the experiments was the simple left-to-right with self state transitions and no jumps were allowed. The initial parameters for B in all experiments were set using a uniform distribution. Initial parameters for A and π for the fully ergodic model were set using a uniform distribution. For left-to-right (LTR) models, π was set to 1 for the first state. For the A matrix, self state transition and transitions from state i to $i + 1$ were allowed with equal probabilities (0.5). To achieve more accurate results, several code books with different sizes were constructed using the K-Means [DHS01] algorithm. In order to overcome the problem of initialization of the K-Means, the algorithm was run using 10 different initializations. The performance in all experiments was measured on the test set of the MNIST database.

6.2.2 Studying the effect of number of states

In studying the effect of number of states, two experiments were conducted. The first experiment used HMMs with a left-to-right topology and all models had an equal number of states. The experiment studied the relation between the performance

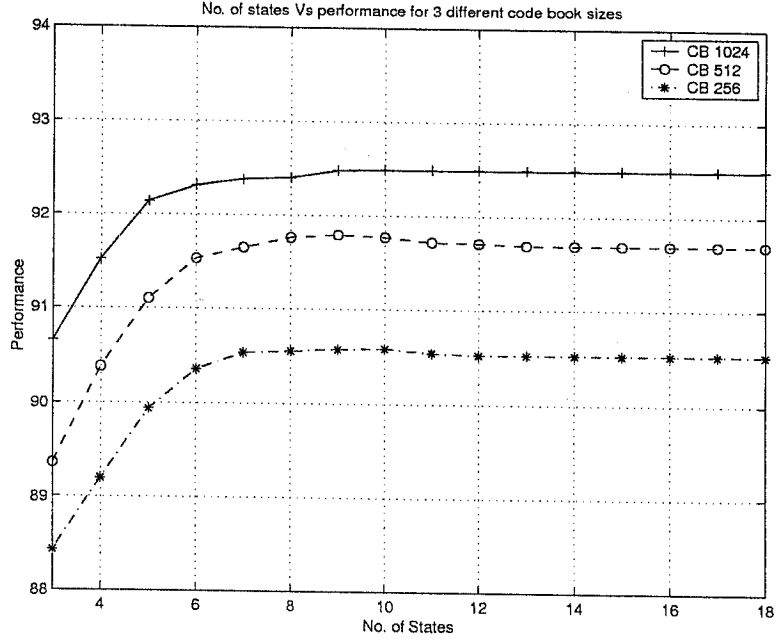


Figure 14: The relation between performance and the number of states with different code book sizes.

and the increase in the number of states in the classifier. The second experiment studied the performance of classifiers with a varying number of states in each model. It compared the performance between models with an equal number of states and models with a varying number of states.

Experiment 1

Figure 14 illustrates the results for experiment 1. It can be seen that increasing the number of states can increase the performance up to a certain limit, and then saturation is reached whenever more unnecessary states are added. However, the saturation may be accompanied by a slight drop in the performance.

The saturation may be explained as follows. The number of states n , is the number of values that the hidden variable can take and accordingly the emission of symbols change. Consider the true (unknown) number of values of the hidden variable is n_0 .

Table 3: Performance measures on the test set as the number of states and varying code book size increased.

<i>No. of States</i>	<i>Code book size</i>		
	256	512	1024
3	88.43	89.36	90.65
4	89.2	90.38	91.52
5	89.94	91.1	92.14
6	90.36	91.53	92.31
7	90.53	91.65	92.38
8	90.55	91.76	92.4
9	90.57	91.79	92.48
10	90.58	91.77	92.49
11	90.54	91.72	92.49
12	90.52	91.71	92.49
13	90.52	91.69	92.49
14	90.52	91.69	92.49
15	90.52	91.69	92.49
16	90.52	91.69	92.49
17	90.52	91.69	92.49
18	90.52	91.69	92.49

If $n \ll n_0$ the model will have a poor modelling and hence a classifier with poor performance. If $n \gg n_0$, additional states will introduce redundancy with no effect on the modelling capability and hence the performance is saturated. Adding more unnecessary states increases the complexity (time and computation) with no effect on the performance.

Experiment 2

The goal of the experiment was to measure the performance of classifiers with different number of states in each model to see how comparable they are with classifiers having all models with an equal number of states. Two HMM classifiers were used. According to the previous experiments, the first classifier had 10 states per model, the second classifier had a different number of states in each model. Determining the number of states in each model will be described in the next subsection. Figure 15 illustrates

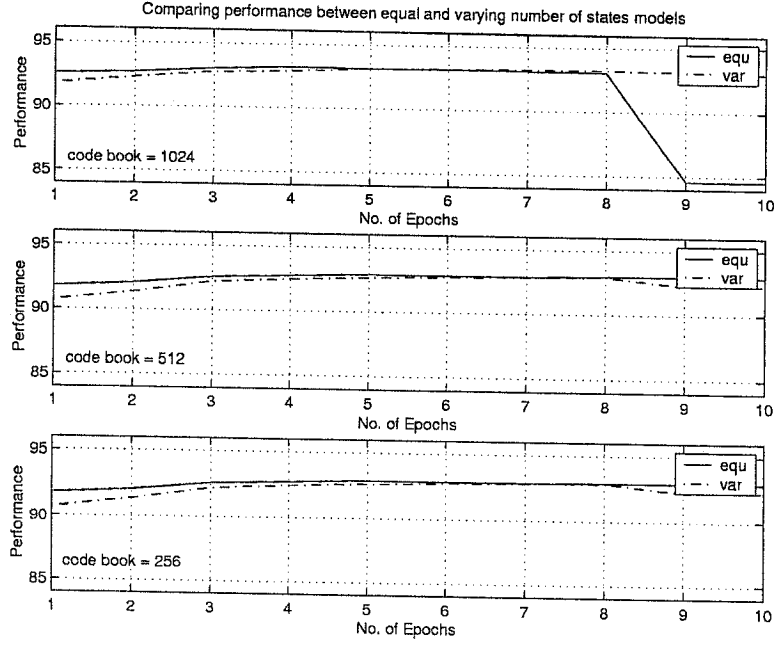


Figure 15: Performance comparison between models with equal (EQU) and varying (VAR) number of states with different codebook sizes.

the results of this experiment. Models with an equal number of states are referred as (EQU) and models with a varying number of states are referred as (VAR). Figure 15 shows clearly how models with a varying number of states can achieve almost the same performance of models with an equal number of states with the advantage of smaller number of states but at the price of more epochs. The total number of states in the EQU models is 100, and the total for VAR models is 70 states. Achieving the same performance with a smaller number of states brings considerable reduction in complexity when it comes to large classification problems. However, as followed in the literature [EYGSS99, ABKP98], a guaranteed performance with an easy design would be an HMM classifier with an equal number of states for all models. In Figure 15, it is worth mentioning that the drop seen in the first graph is experienced in the other graphs for the EQU and VAR models but in late epochs not shown in the graphs. The reason for the drop is due to the overfit of models on the data and due to the diffusion of credits while learning.

Table 4: Comparison between performances of EQU models and VAR models at each epoch on the test set using codebook size = 1024

<i>No. of epochs</i>	EQU	VAR
1	92.49	91.75
2	92.65	92.25
3	92.98	92.71
4	93.19	92.85
5	93.10	93.11
6	93.14	93.16
7	93.08	93.21
8	93.03	93.22
9	84.62	93.13
10	84.57	93.03

Table 5: Comparison between performances of EQU models and VAR models at each epoch on the test set using codebook size = 512

<i>No. of epochs</i>	EQU	VAR
1	91.77	90.66
2	92.06	91.36
3	92.58	92.21
4	92.75	92.47
5	92.92	92.65
6	92.88	92.78
7	92.88	92.86
8	92.97	92.93
9	92.96	92.27
10	92.92	92.31

Table 6: Comparison between performances of EQU models and VAR models at each epoch on the test set using codebook size = 256

<i>No. of epochs</i>	EQU	VAR
1	90.58	90.57
2	91.3	91.11
3	91.87	91.15
4	92.26	91.46
5	92.37	91.83
6	92.38	92.11
7	92.43	92.36
8	92.49	92.53
9	92.48	92.61
10	92.53	92.77

Determining the number of states

As mentioned earlier, the number of states is usually fixed (manually predetermined). An exception are models that use an automatic clustering algorithms that determine the number of states and their outputs, but this still leaves out the topology [Bra96, TK99]. Clustering sequential data while neglecting the variations of the time factor, tends to discover the underlying structure of the data given that the number of clusters is known. To determine the number of states using clustering, we proposed the use a cluster validity index [BP98] to measure the goodness of different clustering configurations and then select the best number of clusters according to this cluster validity index.

In the experiments, the K-Means algorithm [DHS01] was used to cluster the sequential data of each model. The sequential data were portioned from 3 up to 9 clusters and to overcome the problem of initialization of the K-means, the algorithm was run using 10 different initializations. For each clustering configuration, the DB-index [BP98] was used to measure the goodness of clustering. According to the DB-index measure, the number of states (clusters) in each model was determined according to the clustering configuration corresponding to the lowest value of the DB-index. Table

Table 7: The number of states of each model.

<i>Model</i>	0	1	2	3	4	5	6	7	8	9
<i>No. of States</i>	6	5	8	8	9	6	8	8	3	9

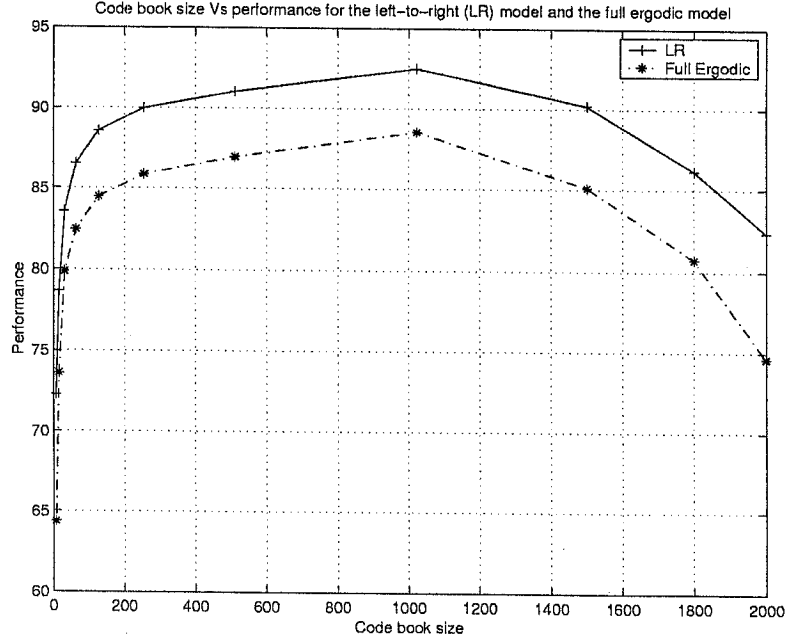


Figure 16: Performance comparison between full ergodic and left-to-right models with different codebook sizes.

7 shows the number of states for each model.

6.2.3 Studying the effect of model topology

To study the effect of the model topology on the performance, two HMM classifiers were considered. Both classifiers had the same number of models and the same number of states in each model but the model topology was different in both classifiers. The first classifier had full ergodic (fully connected) models while the second had left-to-right topology as described earlier. The comparison in performance was measured with different code book sizes. Figure 16 illustrates the results obtained from this experiment. As expected, the results show that the simpler model; which is the left-to-right in that case, always outperforms the full ergodic model. The full ergodic model

Table 8: Performance measures obtained from full ergodic and left-to-right models with different code book sizes.

<i>Code book size</i>	<i>left-to-right (%)</i>	<i>full ergodic (%)</i>
8	72.3	64.36
16	78.71	73.62
32	83.54	79.88
64	86.52	82.49
128	88.57	84.46
256	89.98	85.86
512	91.02	86.97
1024	92.18	88.54
1500	90.21	85.11
1800	86.17	80.72
2000	82.34	74.63

represents a fully connected graph and hence has the largest number of parameters. According to the Bayesian approach, the model has the highest likelihood of the data which led the model to overfit the training set and hence the poor performance on the test set. As for the diffusion of credits factor, the A matrix for the full ergodic model does not have deterministic (0 or 1 probabilities) transitions which made it difficult for the model to learn long range dependencies.

6.3 Experimental Results for the Proposed Framework

We designed a simple prototype for the proposed framework to conduct some preliminary experiments on the recognition of unconstrained handwritten digits. The experiments described in this section shows the results of the proposed framework on the time-series data obtained from the MNIST database. In these experiments, the test set was divided into two sets, a validation set and a test set. The first 5000 samples were taken as the validation set and the other 5000 samples were considered

as the test set.

The HMM based classifier of the modelling stage consisted of 10 discrete HMMs. Each model had 10 states with a simple left-to-right topology with self-state transition. Like the previous experiments, three codebooks of sizes 1024, 512 and 256 were used in these experiments. The discrete HMM-based classifier was trained using the Baum-Welch algorithm until a minimum error rate could be achieved on the validation set. After training, the training set \mathcal{Z} was mapped to the set \mathcal{X} as mentioned in Section (5.3). Since the output probabilities of the models are usually very small, the negative logarithms of the output probabilities were stored instead. Table 9 shows the results obtained from the HMM based classifier on the validation set and the test set with different codebooks.

Table 9: HMM based classifier performance on the validation set and the test set with different codebooks

<i>CodeBook size</i>	<i>Validation set (%)</i>	<i>Test set (%)</i>
256	93.95	93.53
512	94.21	93.97
1024	94.87	94.19

For the classification stage, the package of *SVM^{Light}* V 5.00 [Joa99a] was used as the discriminative classifier. The stage consisted of 10 SVM classifiers (one against all strategy) with a Gaussian kernel. The constant parameter C of the kernel was fixed at 10 and the gamma parameter [Joa99a] of the kernel was adjusted until the minimum error rate could be achieved on the validation set. Table 10 shows the final results obtained from the framework after using the SVM classifiers, and Table 11 shows a comparison between the results obtained from the HMM-based classifier and the proposed framework.

It can be seen from Table 11 how the framework significantly boosted the results

Table 10: The proposed framework performance on the validation set and the test set with different codebooks in the modelling stage

<i>CodeBook size</i>	<i>Validation set (%)</i>	<i>Test set (%)</i>
256	98.53	97.8
512	98.77	97.89
1024	99.07	98.02

Table 11: Comparison of performance of the HMM based classifier and the proposed framework

<i>CodeBook size</i>	<i>HMM based classifier (%)</i>	<i>Proposed framework (%)</i>
256	93.53	97.8
512	93.97	97.89
1024	94.19	98.02

Table 12: Digit error rate for the proposed framework

<i>Digit</i>	<i>No. errors</i>	<i>Error rate (%)</i>
0	5	4.85
1	3	2.7
2	11	10.79
3	11	11.22
4	11	10.9
5	9	9.17
6	7	7.12
7	11	11.0
8	18	18.66
9	13	13.59

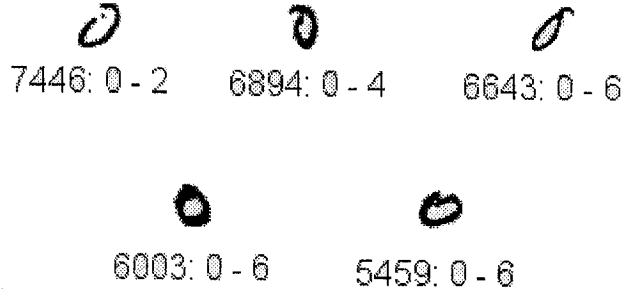


Figure 17: Confusions among digit 0 and other digits

of the standard HMM-based classifier. However, to better understand the performance of the framework, the 1.98% error rate (99 errors) was further analyzed in Tables 12 and 13. Table 12 shows the contribution of each digit to the errors of the whole framework, and for better interpretation of the errors of each digit, the confusion matrix in Table 13 illustrates all the confusions among the digits. Tables are further supported by Figures 17 through 26 that show all the misrecognized digits (99 errors) where each figure shows the confusions encountered among other digits. Each misrecognized digit is labeled in the form $id : x - y$ where id is the digit serial number in the MNIST database, x is the true label and y is the classifier output.

Table 13: Confusion matrix for the errors generated by the proposed framework

	0	1	2	3	4	5	6	7	8	9
0		0	1	0	1	0	3	0	0	0
1	0		1	0	0	0	1	0	1	0
2	1	0		2	1	0	1	4	2	0
3	0	1	3		0	3	0	2	2	0
4	1	2	1	0		1	1	0	0	5
5	1	1	0	2	1		3	0	1	0
6	4	1	0	0	1	1		0	0	0
7	0	2	3	0	2	0	0		1	3
8	1	3	1	3	1	3	1	2		3
9	0	2	0	1	5	1	0	3	1	

By analyzing the confusion matrix, several reasons can be conjectured regarding




 6883: 1 - 6
  6783: 1 - 2
  5331: 1 - 8

Figure 18: Confusions among digit 1 and other digits












 8094: 2 - 8
  8320: 2 - 7
  9839: 2 - 7
  9937: 2 - 8
  5087: 2 - 6
 7724: 2 - 3
  6744: 2 - 4
  6554: 2 - 7
  5820: 2 - 7
  5449: 2 - 0
 9980: 2 - 3

Figure 19: Confusions among digit 2 and other digits












 9905: 3 - 7
  5623: 3 - 2
  5841: 3 - 5
  6065: 3 - 1
  6535: 3 - 5
 6575: 3 - 5
  6722: 3 - 2
  7097: 3 - 7
  8247: 3 - 8
  9173: 3 - 2
 5046: 3 - 8

Figure 20: Confusions among digit 3 and other digits

~~4~~ ~~4~~ ~~4~~ ~~4~~ ~~4~~
 5200: 4 - 9 5440: 4 - 9 6019: 4 - 1 6553: 4 - 2 7434: 4 - 5

~~4~~ ~~4~~ ~~4~~ ~~4~~ ~~4~~
 8520: 4 - 9 9208: 4 - 9 5159: 4 - 9 9906: 4 - 1 8081: 4 - 6

~~4~~
 8107: 4 - 0

Figure 21: Confusions among digit 4 and other digits

~~5~~ ~~5~~ ~~5~~ ~~5~~
 9770: 5 - 0 6042: 5 - 4 8082: 5 - 6 8502: 5 - 8

~~5~~ ~~5~~ ~~5~~ ~~5~~
 5769: 5 - 6 5982: 5 - 3 5735: 5 - 1 5937: 5 - 3

~~5~~
 5098: 5 - 6

Figure 22: Confusions among digit 5 and other digits

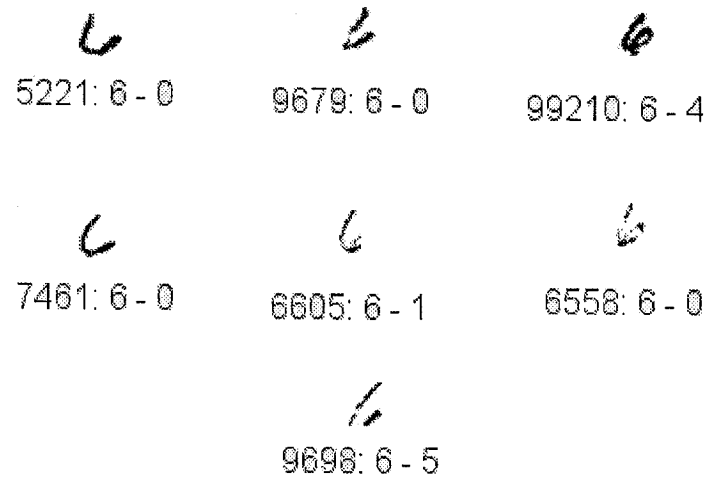


Figure 23: Confusions among digit 6 and other digits

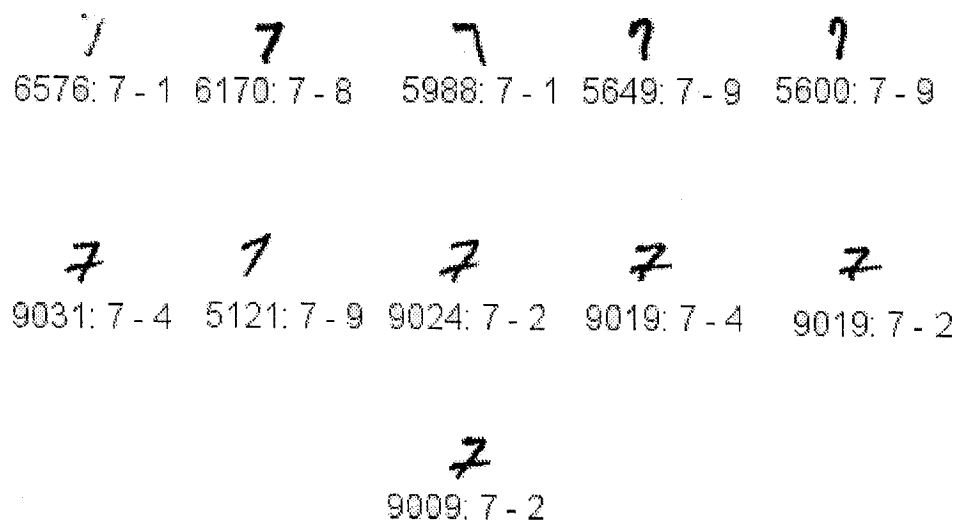


Figure 24: Confusions among digit 7 and other digits



Figure 25: Confusions among digit 8 and other digits

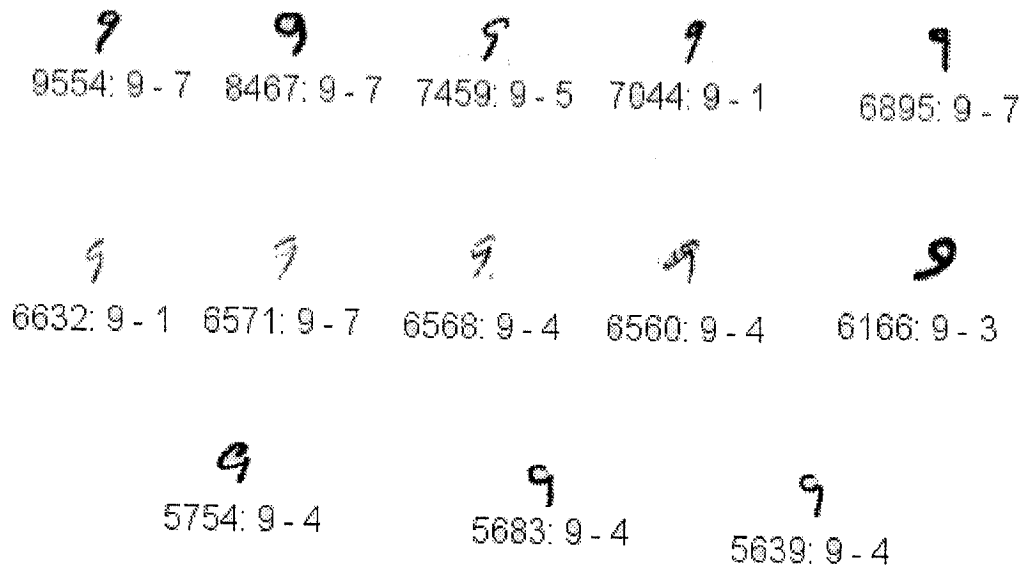


Figure 26: Confusions among digit 9 and other digits

the errors. First, the confusion between digit pairs such as (2,7), (3,2), (3,5), (4,9), (7,2), (7,9), (8,3), (8,5), (8,9), (9,4) and (9,7) can be due to the closeness of the likelihoods generated from each model, the correct and the false models, which is a direct result of the training method. Training is problem independent, and as mentioned earlier, maximizing the likelihood does not guarantee less errors for HMM-based classifiers, and hence a discriminative training can improve the performance of HMMs, and consequently the framework. A second reason would be the type of features used in the experiments. Features are heavily problem dependent and the simple type of features used in the experiments, are not discriminatory enough. Lately, for handwritten digit recognition, Britto *et al.* proposed a set of complex features for HMMs that showed a promising performance on HMM-based classifiers [SdBS03]. Another reason for the error is the likelihood score. In spite of the increase in performance the likelihood score has provided, yet it is not known if there are other fixed size feature vectors that can be extracted from HMMs and yield a better discrimination. What was achieved so far, is that mapping the time-series data to a fixed size vector is a promising approach, but what is the best feature vector that can be extracted, this is still an open question that will be discussed in the next chapter.

In order to evaluate the proposed approach, our results were compared with state-of-the-art recognition results (see Table 14) on the MNIST database [LNSF03, SSP03]. The experienced reader will quickly notice that the comparison might not be fair since our results are based on a smaller test set (5000 samples), however, considering the experimental conditions, our goal was to find out how far these results stand from well known results on the same database. As a future work, an interesting experiment would be to extract the validation set from the training set and repeat all the experiments with all necessary adjustments based on the new validation set.

By comparing the results, one can see that the proposed approach is still far away

Table 14: Comparison of performances of different classifiers on the MNIST database

<i>Classifier</i>	<i>Error rate (%)</i>
<i>Discrete HMMs, codebook 1024 (modelling stage only)</i>	5.81
<i>Proposed framework</i>	1.98
LeNet-4 [LJB ⁺ 95]	1.1
MultiNet Framework [DKS01]	0.99
LeNet-5 [LJB ⁺ 95]	0.95
Boosted LeNet-4 [LJB ⁺ 95]	0.7
Belongie et al. [BMP02]	0.63
Teow et al. [TL02]	0.59
Simard et al. [SSP03]	0.4
VSVM [Don03]	0.38

from the lowest error rate on the database, however, the framework made a jump of 3.83% from the standard HMM results. Moreover, as observed from state-of-the-art techniques and the results presented by Liu and Fujisawa [LNSF03], and Simard *et al.* [SSP03], that the first barrier in a good character recognition system is to achieve around 1.5% on the error rate. By injecting more salient features and classifier tuning, one can achieve the 0.7% toward 0.4%, which is the actual best result. In spite of being a prototype model, the preliminary experiments show the potential of the proposed framework in specific and the potential of generative-discriminative trend in general. Table 14 shows state-of-art recognition results on the MNIST database sorted in descending order according to the error rate.

Chapter 7

Conclusions and Future Work

High performance classification of time-series data is still a challenging problem due to the time variability factor, their variable length and the absence of the priori knowledge in many applications that use HMMs. Despite state of the art results achieved in applications such as speech recognition and handwriting recognition, such applications do not depend solely on HMMs, but priori knowledge from the application domain such as word networks and language models, which have been plugged into the training and testing phase of HMMs and hence the high performance is obtained. Consequently, the more the proposed solution is less dependent on priori knowledge, the better it can fit into many applications. The great ability of HMMs to model time-series data and the ability of generative models, in general, to approximate the true density of the data, combined with the guaranteed classification performance of discriminative models, draw a very promising trend toward a general solution for the problem.

In this thesis, we targeted the low performance of HMM classifiers. First we investigated the effect of the topology and the number of states, each separately, on the performance of HMMs. The investigation, supported by our experimental results showed that the topology has a stronger contribution to the classification performance

of HMM-based classifiers than the number of states. Next, we proposed a new framework that combines generative and discriminative models for classifying time-series data. The two-stage framework uses HMMs in the first stage to model the time-series data and then extracts a static fixed size feature vector (the likelihood score) from all the HMMs. In the second stage, discriminative models are used to classify the likelihood score. The framework was able to improve the results of standard HMMs by 3.83% when tested on the problem of unconstrained handwritten digits recognition. Such an increase shows the potential of the generative-discriminative approach and further research directions for this trend.

Future work : Based on the work presented in this thesis, we discuss several issues that can improve the proposed framework:

Type of application and the data set: The first HMM appeared for speech recognition problems and since that time, the state of the art techniques and results were reported in the speech recognition community. Nowadays, many standard speech databases are available on the World Wide Web. Speech data has several advantages over any other type of time-series data. 1) Speech is a one dimensional signal that by nature is in time-series format. 2) Unlike any other data, the speech recognition community was able to standardize the type of features extracted from speech data. Therefore researchers do not have to worry about issues like best discriminative features.

Unlike speech recognition, handwriting recognition applications still face two main challenges; first, the 2-dimensional signal representation of an image, and second, the type of features to be extracted from the image. As for the first challenge, yet there is no standard method to model a 2-D signal into a 1-D signal like speech. For the second, to the best of our knowledge and despite many excellent results of handwriting recognition systems, yet there are no

standard features extracted from binary or gray level images in the form of time-series data. Therefore, it will be of great importance to standardize the modelling method of images using HMMs, in order to direct the research effort to more advanced problems.

Finally, in our experiments, we used images of handwritten digits that have a fixed width and height and hence all sequential data extracted from the images had the same length. Usually digital images for words and speech data exist in variable length sequential data, therefore it will be interesting to study this effect on the performance of the proposed framework.

Type of HMMs: The original theory of HMMs was based on continuous density HMMs which are richer models than discrete ones. For implementation issues of continuous density HMMs, a very well known ready tool such as HTK (HMMs Tool Kit) can be used to save time and effort to develop continuous density HMMs.

The score: In this thesis, we proposed the use of the likelihood score extracted from the HMMs of the generative stage. It is worth trying to find out how comparable is the Fisher score to the likelihood score. Also, instead of using the likelihood score, which is the summation of all possible paths in the alpha computation [Rab89], the non-zero elements of the last row of the matrix alpha, $\alpha_T(i)$ for all states i can be used instead. This vector can be augmented with other vectors from other HMMs and then classified using the discriminative model. The feature vector could be more representative than the likelihood score since it contains information on each state inside the model.

Increasing discrimination between models: As mentioned earlier, increasing the discrimination between HMMs will increase the performance of the classification stage. This can be achieved by optimizing the structure of each model in the classifier using Stolcke's algorithm [SO92], and using parameter estimation algorithms that interacts with models of other classes such as the MCE [JK92, KL98, SR00, GH02, Bie02] algorithm.

MLPs Vs SVMs: SVMs are known for their good generalization in classification problems, however the testing phase of SVMs is still very slow to be used in real life applications such as speech recognition or handwriting recognition. Speeding up the testing phase of SVMs is currently a hot research topic in the machine learning community. For real life applications, MLPs represent the alternative with several advantages. First, MLPs have a very fast testing phase when compared with SVMs speed. Second, the output of MLPs can be considered as a posterior probability which can be easier to combine with the likelihood output of HMMs than to combine with the normalized distance obtained from SVMs. Moreover, by looking at Table 14, most of the state of the art results on MNIST database were obtained using neural network classifiers.

Open questions: In our approach, we reduced the role of HMMs from modelling and classification to modelling only and let discriminative models be in charge of classification. This is done by extracting a fixed size feature vector from the HMMs, then using a discriminative model to classify these feature vectors. An interesting question, what would be the feature vector extracted from the HMMs that best describes the time-series pattern? Should it be extracted from a single HMM, (the class of the pattern only), or should it be from all HMMs, (more toward a majority voting)?

A second question; as in static data, there is the notion and the concept of

the mean vector and the covariance matrix. How can these two terms can be explained mathematically in the case of variable length time-series data?

Third, considering that our approach is a kind of voting scheme, how can this scheme fit other classifiers? Should they always be a, generative discriminative pair? Or can it be a discriminative discriminative pair?

Bibliography

- [ABKP98] E. Augustin, O. Baret, S. Knerr, and D. Price. Hidden Markov model based word recognition and its application to legal amount recognition of French bank cheques. *Computer Vision and Image Understanding*, 70(3):404–419, 1998.
- [ACS02a] N. E. Ayat, M. Cheriet, and C. Y. Suen. Kmod a two parameter svm kernel for pattern recognition. In *Proc. of 16th ICPR, Quebec city, Canada*, pages 200–204, 2002.
- [ACS02b] N-E. Ayat, M. Cheriet, and C. Y. Suen. Optimzation of the SVM kernels using an empirical error minimization scheme. In S-W Lee and A. Verri, editors, *Pattern Recognition with Support Vector Machines*, pages 354–369. Springer, 2002.
- [Bak75] J. K. Baker. The dragon system - an overview. *IEEE Trans. Accoustics, Speech and Signal Proc.*, 23(11):23–29, 1975.
- [Bak76] R. Bakis. Continuous speech recognition via centisecond acoustic states. In *The 91st Meeting of the Acoustical Society of America*, 1976.
- [Bal93] V. Balasubramanian. Equivalence and reduction of hidden Markov models. Technical Report 1370, MIT Aritifical Intelligence Laboratory, 1993.

- [BBdSM86] L. Bahl, P. Brown, P. de Souza, and R. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. of ICASSP, Tokyo*, pages 49–52, 1986.
- [BBdSM88] L. Bahl, P. Brown, P. de Souza, and R. Mercer. A new algorithm for the estimation of hidden Markov model parameters. In *Proc. of ICASSP*, pages 493–497, 1988.
- [BBL01] C. Bahlman, H. Burkhardt, and A. Ludwigs. Measuring HMM similarity with the Bayes probability of error and its application. In *Proc. of 6th ICDAR, Seattle, Washington, USA*, pages 406–411, 2001.
- [BDM01] M. Bicego, A. Dovier, and V. Murino. Designing the minimal structure hidden Markov model by bisimulation. In M. Figueiredo, J. Zerubia, and A. K. Jain, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 75–90. Springer, 2001.
- [Ben99] Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 41(1):129–162, 1999.
- [BF95] Y. Bengio and P. Frasconi. Diffusion of credits in Markovain model. In *Proc. of Advances in Neural Information Procesing Systems 7*, pages 1251–1254, 1995.
- [BF96] Y. Bengio and P. Frasconi. Input/Output HMMs for sequence processing. *IEEE Trans. on Neural Networks*, 7(5):1231–1249, 1996.
- [BH01] A. Brown and G. Hinton. Products of hidden Markov models. In T. Jaakkola and T. Richardson, editors, *Proc. of Artificial Intelligence and Statistics*, pages 3–11. 2001.

- [Bie02] A. Biem. Minimum classification error training for online handwritten word recognition. In *Proc of 8th IWFHR, Niagra-on-the-lake*, pages 61–65, 2002.
- [Bie03] A. Biem. A model selection criterion for classification: Application to HMM topology optimization. In *Proc. 17th ICDAR, Edinburgh, U.K*, pages 104–108, 2003.
- [BMF03] M. Bicego, V. Murino, and M. Figueiredo. A sequential pruning strategy for the selection of the number of states in hidden Markov models. *Pattern Recognition Letters*, 24:1395–1407, 2003.
- [BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, 24(4):509–522, 2002.
- [Bou03] G. Bouchard. The trade-off between generative and discriminative classifiers. In *Proc. of Advances in Neural Information Processing 16*, 2003.
- [BP66] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.*, 37:1554–1563, 1966.
- [BP70] L. E. Baum and T. Petrie. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. of Math. and Stat.*, 41(1):164–171, 1970.
- [BP98] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Trans. on Sys. Man and Cybernetics Part B*, 28(3):301–315, 1998.
- [BR99] C. Becchetti and Lucio Prina Ricotti. *Speech Recognition, Theory and C++ Implementation*. John Wiley & Sons, West Sussex, England, 1999.

- [Bra96] Thorsten Brants. Estimating markov model structures. In *Proceedings of the Fourth Conference on Spoken Language Processing (ICSLP-96)*, Philadelphia, PA, 1996.
- [Cor96] S. Cornell. *A Comparison of Hidden Markov Model Features for the Recognition of Cursive Handwriting*. PhD thesis, Dept. of Computer Science, Michigan State University, 1996.
- [CST00] N. Cristianin and J. Shawe-Taylor. *An Introduction to support vector machines and other kernel-based learning methods*. Cambridge Univ. Press, Cambridge, England, 2000.
- [DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, Second Edition*. Wiley-Interscience, Canada, 2001.
- [DKS01] J.X. Dong, A. Krzyzak, and C. Y. Suen. A multi-net learning framework for pattern recognition. In *Proc. of 16th ICDAR, Seattle, USA*, pages 255–268, 2001.
- [DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Royal Statistics Soc.*, 39:1–38, 1977.
- [Don03] J. Dong. *Speed and accuracy, Large-scale machine learning algorithms and their applications*. PhD thesis, CENPARMI, Department of Computer Science, Concordia University, Montreal, Canada, 2003.
- [EYGSS99] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen. An HMM based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans. PAMI.*, 21(8):752–760, 1999.

- [GH02] Y. Ge and Q. Huo. A study on the use of CDHMM for large vocabulary offline recognition of handwritten Chinese character. In *Proc. of 18th IWFHR, Niagra-on-the-lake, Canada*, pages 334–338, 2002.
- [Gha97] Z. Ghahramani. Learning dynamic bayesian networks. In C. Gile and M. Gori, editors, *Lecture Notes in Artificial Intelligence: Adaptive Processing of Temporal Information*. 1997.
- [GJ97] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–275, 1997.
- [GL92] J-L. Gauvain and C-H. Lee. Map estimation of continuous density HMM: Theory and applications. In *Proc. of DARPA Speech & Nat. Lang. Processing*, Feb. 1992.
- [GSSG00] F. Grandidier, R. Sabourin, C. Y. Suen, and M. Gilloux. A new strategy for improving feature sets in a discrete HMM-based handwritten word recognition system. In *Proc of 7th IWFHR, Amsterdam*, pages 113–122, 2000.
- [GZL01] G. Guo, H. Zhang, and S. Li. Support vector machines for face recognition. *Imag. Vis. Comput.*, 19(9 & 10):631–638, 2001.
- [Hec96] D. Heckerman. A tutorial on learning with graphical models. Technical Report MSR-TR-9506, Microsoft Research, 1996.
- [Jel72] F. Jelinek. Continuous speech recognition by statistical methods. *Proc. of IEEE*, 64:532–556, 1972.
- [JH98] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. of Advances in Neural Information Processing 11*, 1998.

- [JK92] B-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. Signal Processing*, 40(12):3043–3054, 1992.
- [Joa99a] T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.
- [Joa99b] T. Joachims. Text categorization with support vector machines. In *Proc. Int. Conf. Machine Learning*, pages 137–142, 1999.
- [JR79] B-H. Juang and L. Rabiner. The segmental k-means algorithm for estimating parameters of hidden Markov models. *IEEE Trans. on Speech, Acoustics and Signal Processing*, 38(9):1639–1641, 1979.
- [KL98] S. Katagiri and C-H. Lee. Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method. *Proc. of the IEEE*, 86(11):2345–2373, 1998.
- [Koh97] T. Kohonen. *Self-Organizing Maps*. Springer, 1997.
- [KRSG02] A. Koerich, C. Y. Suen R. Sabourin, and M. Gilloux. A hybrid large vocabulary handwritten word recognition system using neural networks with hidden Markov models. *Proc. of 8th IWFHR, Niagra-on-the-lake*, pages 99–104, 2002.
- [LeC] Y. LeCun. The MNIST database of handwritten digits. <http://yann.lecun.com/exdb/mnist>.
- [Lee99] K-F. Lee. *Automatic Speech Recognition: The development of the SPHINX System*. Kluwer Academic Press, 1999.

- [LJB⁺95] Y. LeCun, L. Jackel, L. Bottou, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V.N. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *Proc. Intl. Conf. Artificial Neural Networks*, pages 53–60, 1995.
- [LNSF03] C-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: Benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36:2271–2285, 2003.
- [LPhN99] R. Lyngso, C. Pedersen, and h. Nielsen. Metrics and similarity measures for hidden Markov models. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, pages 178–186, 1999.
- [Mur01] K. P. Murphy. A introduction to graphical models. <http://www.ai.mit.edu/~murphyk/papers.html>, 2001.
- [NG02] J. Ng and S. Gong. Composite support vector machines for detection of faces across views and pose estimation. *Imag. Vis. Comput.*, 20(5 & 6):359–368, 2002.
- [NJ02] A. Ng and M. Jordan. On generative vs. discriminative classifiers: A comparison of logistic regression and naive bayes. In *Proc. of Advances in Neural Information Processing 15*, 2002.
- [PCST00] J. Platt, N. Critianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Proc. of Advances in Neural Information Processing systems 13*, pages 547–553, 2000.
- [PMSM⁺03] L. Prevost, C. Michel-Sendis, A. Moises, L. Oudut, and M. Millgram. Combining model-based and discriminative classifiers: Application to

- handwritten character recognition. In *Proc. of 7th ICDAR, Edinburgh*, pages 31–35, 2003.
- [QB02] L. Quan and S. Bengio. Hybrid generative-discriminative models for speech and speaker recognition. Technical report, IDIAP, March 2002.
- [Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected application in speech recognition. *Proc. of IEEE*, 77(2):257–286, 1989.
- [Rai86] J. Raissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14, 1986.
- [RH97] Y. Rubenstein and T. Hastie. Discriminative vs informative learning. In *Proc. of Knowledge Discovery and Data Mining*, 1997.
- [Row99] S. Roweis. Constrained hidden Markov models. In *Proc. of Neural Information Processing, 12*, pages 782–788, 1999.
- [SdBS03] F. Bortolozzi S. de Britto, R. Sabourin and C. Y. Suen. Complimentary features combined in an HMM-based system to recognize handwritten digits. In *Proc. of 12th Intl. Conf. on Image Analysis and Processing*, pages 670–675, 2003.
- [Sen86] E. Senta. *Nonnegative Matrices and Markov Chains*. Springer, New Yor, 1986.
- [SG01] E. Santos and H. Gomes. Appearance-based object recognition using support vector machines. In *Proc. of XIV Brazilian Symp. Computer Graphics and Image Processing*, pages 399–405, 2001.
- [Shw] G. Shwarz. Estimating the dimensionality of a model. *The Annals of Statistics*, 6(2):461–464.

- [SO92] A. Stolcke and S. Omuhundro. Hidden Markov model induction by Bayesian model merging. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing 5*, pages 11–18. Morgan Kaufmann, 1992.
- [SR00] L. Saul and M. Rahim. Maximum likelihood and minimum classification error rate factor analysis for automatic speech recognition. 8(2):115–125, 2000.
- [SSP03] P. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. of 17th ICDAR, Edinburgh, U.K*, pages 962–965, 2003.
- [SW96] Y. Singer and M. Warmuth. Training algorithms for hidden Markov models using entropy based distance functions. In *Proc. of Advances in Neural Information Processing Systems 9*, 1996.
- [TK99] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*, chapter 9. Academic Press, San Diego, 1999.
- [TL02] L-N. Teow and K-F. Loe. Robust vision-based and classification schemes for off-line handwritten digit recognition. *Pattern Recognition*, 35(11):2355–2364, 2002.
- [Vap98] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Sussex, England, 1998.
- [Vit83] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE. Trans. Information Theory*, 13(4):179–190, 1983.

- [WC00] V. Wan and W. Campbell. Support vector machines for speaker verification and identification. In *Proc. of IEEE Workshop on Neural Networks for Signal Processing*, pages 775–784, 2000.
- [WCH02] Y. Wang, C. Chua, and Y. Ho. Facial feature detection and face recognition from 2D and 3D images. *Patt. Recog. Lett.*, 23(10):1191–1202, 2002.
- [ZLX00] B. Zhao, Y. Liu, and S. Xia. Support vector machines and its application in handwritten numerical recognition. In *Proc. of 15th ICPR, Barcelona, Spain*, pages 720–723, 2000.
- [ZLZ01] L. Zhang, F. Lin, and B. Zhang. Support vector machine learning for image retrieval. In *Proc. IEEE Int. Conf. Image Processing*, pages 721–724, 2001.