Technical Report

UCAM-CL-TR-652
ISSN 1476-2986

Number 652

UNIVERSITY OF
CAMBRIDGE

Computer Laboratory

# Discriminative training methods and their applications to handwriting recognition

Roongroj Nopsuwanchai

November 2005

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

*http://www.cl.cam.ac.uk/*

© 2005 Roongroj Nopsuwanchai

This technical report is based on a dissertation submitted August 2004 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Downing College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

*http://www.cl.cam.ac.uk/TechReports/*

ISSN 1476-2986

# Summary

This thesis aims to improve the performance of handwriting recognition systems by introducing the use of discriminative training methods. Discriminative training methods use data from all competing classes when training the recogniser for each class. We develop discriminative training methods for two popular classifiers: Hidden Markov Models (HMMs) and a prototype-based classifier. At the expense of additional computations in the training process, discriminative training has demonstrated significant improvements in recognition accuracies from the classifiers that are not discriminatively optimised. Our studies focus on isolated character recognition problems with an emphasis on, but not limited to, off-line handwritten Thai characters.

The thesis is organised as followed. First, we develop an HMM-based classifier that employs a Maximum Mutual Information (MMI) discriminative training criterion. HMMs have an increasing number of applications to character recognition in which they are usually trained by Maximum Likelihood (ML) using the Baum-Welch algorithm. However, ML training does not take into account the data of other competing categories, and thus is considered non-discriminative. By contrast, MMI provides an alternative training method with the aim of maximising the mutual information between the data and their correct categories. One of our studies highlights the efficiency of MMI training that improves the recognition results from ML training, despite being applied to a highly constrained system (tied-mixture density HMMs). Various aspects of MMI training are investigated, including its optimisation algorithms and a set of optimised parameters that yields maximum discriminabilities.

Second, a system for Thai handwriting recognition based on HMMs and MMI training is introduced. In addition, novel feature extraction methods using block-based PCA and composite images are proposed and evaluated. A technique to improve generalisation of the MMI-trained systems and the use of $N$-best lists to efficiently compute the probabilities are described. By applying these techniques, the results from extensive experiments are compelling, showing up to 65% relative error reduction, compared to conventional ML training without the proposed features. The best results are comparable to those achieved by other high performance systems.

Finally, we focus on the Prototype-Based Minimum Error Classifier (PBMEC), which uses a discriminative Minimum Classification Error (MCE) training method to generate the prototypes. MCE tries to minimise recognition errors during the training process using data from all classes. Several key findings are revealed, including the setting of smoothing parameters and a proposed clustering method that are more suitable for PBMEC than using the conventional methods. These studies reinforce the effectiveness of discriminative training and are essential as a foundation for its application to the more difficult problem of cursive handwriting recognition.

# Publications

- Roongroj Nopsuwanchai and William F. Clocksin. Hidden Markov Models for Off-line Thai Handwriting Recognition. *In the Proceedings of the 11th International Conference on Artificial Intelligence Applications (ICAIA'03)*, pages 180-189, Cairo, Egypt, February 2003.
- Roongroj Nopsuwanchai and Alain Biem. Discriminative Training of Tied Mixture Density HMMs for Online Handwritten Digit Recognition. *In the Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'03)*, volume 2, pages 817-820, Hong Kong, China, April 2003.
- Roongroj Nopsuwanchai and Dan Povey. Discriminative Training for HMM-Based Offline Handwritten Character Recognition. *In the Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pages 114-118, Edinburgh, Scotland, August 2003.
- Roongroj Nopsuwanchai and Alain Biem. Prototype-Based Minimum Error Classifier for Handwritten Digits Recognition. *In the Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'04)*, volume 5, pages 845-848, Montreal, Canada, May 2004.

# Acknowledgements

This dissertation could not have been completed without the help, support and encouragement from several individuals. First of all, I would like to thank my former supervisor, Prof. William Clocksin, for giving me the opportunity to pursue a PhD, supervising me in my first year, and providing valuable guidance throughout my research and dissertation. I would like to express my deepest gratitude to my current supervisor, Dr. Neil Dodgson, who has motivated me when I am confronted with obstacles and provided an overall guidance and valuable advice on my research direction and English grammar. Equally, I am deeply indebted to Dr. Alain Biem, my mentor at IBM T. J. Watson Research Center, who taught me a theoretical background of discriminative training which significantly helped me to embody the rigorous knowledge in this dissertation. He has constantly provided me with technical guidance, encouragement, support and useful comments. The main experiments in Chapters 4 and 8 were completed under his supervision during my internships at IBM.

I am very grateful to my colleague, Dan Povey, for the provision of several ideas regarding MMI training and the `HMMIRest` re-estimation tool and associated libraries for HTK. The toolkit provided a flexible platform to extend and evaluate the new ideas described in Chapter 6. Many thanks go to the development team of the outstanding HTK toolkit at Cambridge University Engineering Department, which has saved me a lot of time and effort in getting my experiments to work.

I would like to thank everyone in the Embedded Applications Department, IBM T. J. Watson Research Center, for their helpful suggestions and ideas during group meetings. These include Dr. Jane Snowdon (manager of the department who gave me the opportunity to do two summer internships at IBM), Dr. Michael Perrone, Dr. John Pitrelli, Dr. Yi-Min Chee, Dr. Eugene Ratzlaff, Dr. Jianying Hu and Dr. Greg Russell.

I acknowledge the scholarship from the Cambridge Thai Foundation and the Cambridge Overseas Trust. I am greatly in debt of their generous support, without which I would not have a chance of studying in the UK. I also wish to thank Downing College and the Computer Laboratory for funding my attendance at many conferences.

I am very grateful to my friends, Somboon Kulvisaechana (who tirelessly proofread this dissertation and prepared tasty meals at times when I was busy), Leo Boonchin Patanapongpibul (who provided additional computing resources at my critical period) and Panit Watcharawitch (who helped with the LaTeX style files of this dissertation). I also thank for their friendship and many valuable discussions we have had throughout my

PhD journey.

Special thanks go to Sarin Watcharabusaracum and NECTEC for supplying several useful resources regarding Thai OCR and the on-line handwriting database (although the database has not been used in this research), Namfon Assawamekin who provided a number of research papers regarding Thai character recognition, many Thai students at universities in the UK, France and Germany who helped contributing their handwriting samples, and my friends and colleagues, Marco Palomino and Prem Fernando, for their enjoyable company.

Thanks to all my housemates for their friendship and for looking after my well-being during the final period of my research, in particular, Mutita Akusuwan (also for her tutorial on the EM algorithm), Jamorn Somana, Paiboon Sreearunothai, Pasu Sirisalee, Anuphon Laohavisit, Wen Yueh Hsieh and Panida Saikhwan.

This dissertation is dedicated to all my family members, who always support and encourage me through the years, and to my late friend, Meuanjai Ampunsang, who never had a chance to finish her PhD.

# Contents

# List of Figures

# List of Tables

# Part I

# Background

# Chapter 1

# Introduction

## 1.1 Introduction

Handwriting is a natural form of human communications that has existed for centuries. The use of handwriting is involved in many of our day-to-day activities, such as note taking, form filling and letter addressing. During the past two decades, there have been increasing demands for the applications to automatically process the content of these handwritten documents, as a supplement to the tasks performed by humans. In connection with that, handwriting offers an attractive and efficient method to interact with a computer, as witnessed from the recent advent of many hand-held devices that accept handwriting inputs. These devices also require such applications to recognise the handwriting entries. As a result, research in handwriting recognition becomes increasingly important as it provides a solution for these problems.

While the term *character recognition* is generally concerned with the problem of recognising any forms of characters, *handwriting recognition* specifically aims at a more difficult problem of recognising handwritten characters. The ultimate goal of handwriting recognition research is to develop systems that can read any text with the same recognition accuracy as humans, but at a faster rate [2]. Handwriting recognition is a difficult task due to its large problem dimension, which is composed of several sub-problems and the generally ambiguous nature of handwriting itself. There have been an extensive number of handwriting recognition research projects trying to tackle different aspects of the problem. The performance of handwriting recognition systems has improved dramatically over the past decade, especially in some specific tasks, such as handwritten address reading and the recognition of amounts on bank cheques [3]. However, the performance of general handwritten word and sentence recognition is rather low and still not comparable to that achieved by humans.

The focus of this thesis is to improve the overall recognition performance of the handwriting recognition systems by introducing alternative training methods that enhance their ability to discriminate the characters. This chapter aims to introduce essential background to the research (Section 1.2) and the motivation behind the study

(Section 1.3). The research objectives are defined in Section 1.4, followed by the scope of research in Section 1.5. Section 1.6 summarises the key contributions of this thesis. Finally, the outline of this dissertation is presented in Section 1.7.

## 1.2 Research Background

A prerequisite of the character recognition and discriminative training concepts is essential in order to thoroughly understand the motivation behind this research. The following sections broadly explore these core topics and introduce some important terminology.

### 1.2.1 Handwriting Recognition Overview

The problem of handwriting recognition can be classified into two main groups, namely *off-line* and *on-line* recognition, according to the format of handwriting inputs. The off-line recognition systems recognise the characters after they have been written on a piece of paper, scanned into the computer and stored in the image format. On the other hand, the on-line systems can access dynamic information of handwriting strokes while the characters are being written on a tablet or a digitiser. Although the recognition processes of both systems are different in their details, they can be broken down into four fundamental sequential stages: pre-processing, feature extraction, classification and post-processing.

Pre-processing is essential to prepare handwriting inputs to be suitable for later recognition stages. Examples of the pre-processing stage include the processes to identify text regions within the handwritten documents, eliminate imperfections, and segment the texts into isolated characters. The goal of the feature extraction stage is to obtain a compact description (a feature vector) that can be used to uniquely represent the character. Classification is the main decision making stage in which the extracted features are classified into one of several categories. Modern handwriting recognition research is dominated by the use of statistical methods for classification, i.e. statistical classifiers, as seen in a recent survey [4]. Post-processing typically forms a verification step, such as the use of language models and contextual information to verify the recognised characters or words.

A good handwriting recognition system should have the ability to cope with variations in the writing styles while being capable of distinguishing similar yet different characters. All the aforementioned stages are equally important for the recogniser to achieve this property. This thesis focuses mainly on the classification stage in order to improve the performance of the overall system. The appropriate features for the proposed classifiers are also taken into account.

### 1.2.2 Non-Discriminative and Discriminative Methods

In order to classify the unseen samples efficiently, some training algorithms are required to train the classifiers according to the presence of the training samples. We can broadly divide the classifiers into two groups based on their training approaches: *non-discriminative* and *discriminative* classifiers. Although discriminative training is thoroughly explained in Chapter 2, a brief concept of both methods is presented in this section for basic understanding of the subject.

Non-discriminative classifiers, sometimes referred to as generative [5] or informative [6] classifiers, aim at building a model to represent the training samples in each class. Given the unseen character, classification is done by choosing the model that best explains the data. Examples of non-discriminative classifiers are Hidden Markov Models (HMMs) and Gaussian Mixture Models. These classifiers usually rely on non-discriminative training methods such as Maximum Likelihood (ML) estimation in which the model of each class is trained separately by using the samples that belong to the class.

Discriminative classifiers, on the other hand, only centre on the classification decisions, and hence do not attempt to construct a model over the training samples. They focus directly on determining the class decision boundaries, which is equivalent to learning a direct mapping from the training samples to their class labels. Examples include neural networks and Support Vector Machines (SVMs). The training process requires simultaneous considerations of the training samples of all competing classes through the use of discriminative training methods, which involve the optimisation of some discriminative training criteria.

This thesis focuses on applying the *discriminative training criteria* to the *non-discriminative classifiers* with the aim of improving their recognition performance. The motivation for this is described in Section 1.3.

### 1.2.3 Parametric and Non-Parametric Techniques

In statistical pattern recognition, we need to estimate the probability densities from the training samples and use them as if they are the true densities. Various approaches to density estimation can be categorised into two groups: the *parametric* and *non-parametric* techniques. The parametric techniques require an assumption of the correct functional form of the underlying probability density. The training process uses the available training samples to estimate parameters of the function. In contrast, the non-parametric techniques tries to estimate the probability density from the entire sample set without specifying its functional form in advance. In this thesis, we demonstrate the applications of discriminative training to both techniques.

## 1.3 Research Motivation

There are two key motivating factors for this research. These consist of recent advances in handwriting and speech recognition research and the demand for handwriting recognition applications in other non-English languages.

A number of classification techniques in handwriting recognition have been influenced by the speech recognition community. Examples include HMMs, recurrent networks [7] and a hybrid of neural networks and HMMs. The HMMs, in particular, have played an important role in handwriting recognition, as seen from an increasing number of their applications [3]. Training of HMMs in these applications relies on the conventional ML training method, although the recent development in speech recognition has demonstrated an improvement in the recognition performance of the HMMs when discriminative training is applied. We take advantage of this recent development and introduce discriminative training to the handwriting recognition problems. Nevertheless, discriminative training is an ongoing research area, and many of its aspects have not been explored. Some of its important findings are revealed in this research.

Most handwriting recognition research centres around the English or Latin characters, with sparse applications in other languages such as Arabic, Chinese and Hangul. The Thai language (an official language used in Thailand) is among the other Asian languages that receive an increasing demand for such applications, as seen from a dramatic growth of the reports on Thai language processing [8] and the number of computer and Internet users in Thailand [9]. In addition, the Thai characters pose a unique challenge for automatic handwriting recognition on the grounds that there are several groups of Thai characters which have similar shapes, and vary only in small details. In this research, the Thai handwriting recognition problem serves as one of the main applications that is benefited from our proposed methods. Other recognition tasks based on Latin characters are also investigated throughout the dissertation.

## 1.4 Research Objectives

This research introduces the use of discriminative training methods to the handwriting recognition problems, with the aim of improving the recognition performance of the non-discriminatively optimised systems. Three objectives in conjunction with this main goal are summarised as follows:

1. To explore the effectiveness of discriminative training in the HMM-based handwriting recognisers, with an emphasis on the discrimination training criterion called Maximum Mutual Information (MMI) because it has been under-researched in previous work. Some of the important aspects and limitations of MMI training are also investigated, which include its performance under a highly constrained environment (such as the system based on tied-mixture density HMMs, TDHMMs), its optimisation algorithms and the set of optimised parameters that provides maximum discriminabilities.

2. To derive a high-performance Thai handwriting recognition system based on the HMMs and MMI training. This includes an investigation of various feature extraction methods and the introduction to the novel techniques, namely *block-based PCA* and *composite images*. The optimisation algorithm recently proposed for MMI training in speech recognition [10] is evaluated. Other useful techniques for efficiently computing the likelihoods and improving the generalisation of MMI training are also studied.

3. To investigate the use of discriminative training in the prototype-based classifier, which is a representative of the non-parametric techniques, for the handwriting recognition problems. We concentrate on the discriminative training criterion, known as Minimum Classification Error (MCE). The aim is to indicate the importance of various parameter settings in the system, and propose a method to overcome inconsistencies which occur during its training process.

## 1.5 Scope of Research

This research focuses mainly on the classification stage in handwriting recognition (see Section 1.2.1), with an assumption that the handwriting data have been pre-processed and segmented into isolated characters. It is inevitable that the feature extraction process has to be taken into account when designing the recogniser. Most of our experiments rely on the feature extraction methods proposed in the previous literature which yield reasonably effective results on our recognition tasks. Nevertheless, the features for the HMM-based Thai handwriting recognition systems are thoroughly examined due to a lack of their grounded studies. The present study is sufficiently comprehensive to justify the effectiveness of the proposed methods, by comparing the results derived from our systems with those generated by other high-performance classifiers.

In this thesis, we limit our study to the recognition of isolated characters, both on-line and off-line. This can be applied to a wide range of applications such as automatic recognition of the handwritten entry in the forms and recognising the handwriting inputs in the hand-held devices. Thai handwriting is, by its nature, also written in isolation. Moreover, isolated character recognition provides a suitable framework to explore the effectiveness of discriminative training. As soon as we thoroughly understand its properties, we should be able to proceed to a more difficult task of cursive handwriting recognition. It is anticipated that the discriminative training techniques reported in this thesis may be of general use to improve the performance of such a system.

## 1.6 Research Contributions

The main contribution of this thesis is the development of high-performance handwriting recognition systems based on discriminative training methods, which are proven effective and can improve the recognition performance of the non-discriminatively optimised systems. Other significant contributions are presented in Chapters 4, 6, and 8, each of which respectively covers each topic of the research objectives in Section 1.4. These can be briefly summarised as follows:

### 1.6.1 Discriminative Training in HMMs

- MMI training in the HMM-based on-line handwriting recognition is implemented and shows some improvements in the recognition performance when compared to ML training.
- The effectiveness of MMI training in a highly-tied system is demonstrated. The structure of such a system is different from most applications of MMI training in the literature.
- Three optimisation algorithms for MMI training are evaluated.
- A set of the HMM parameters that yields maximum discriminabilities is revealed, providing a guideline for parameter optimisation in MMI training.

### 1.6.2 Thai Handwriting Recognition and MMI Training

- A database of Thai handwritten characters is constructed.
- A high-performance system for Thai handwriting recognition based on the HMMs and MMI training is introduced, and various feature extraction methods are evaluated.
- Novel techniques to improve the recognition performance based on block-based PCA and composite images are introduced.
- An optimisation algorithm proposed by Povey [10] for MMI training is successfully applied to our recognition problem and has demonstrated a fast convergence rate.
- A technique to improve the ability of MMI training to generalise the unseen data and the use of $N$-best lists to efficiently compute the likelihood statistics are presented.

### 1.6.3 Discriminative Training in Prototype-Based Classifiers

- MCE training in the prototype-based classifier, namely the Prototype-Based Minimum Error Classification (PBMEC), is implemented and successfully used in off-line handwriting recognition.
- We reveal that the values of smoothing parameters in the initialisation and MCE training processes of PBMEC have a significant impact on the recognition performance, thereby providing a useful guideline for parameter settings.
- A novel method for prototype initialisation in PBMEC that overcomes an inconsistency between the criteria used in the initialisation and recognition processes is introduced.

- The potential applications of discriminative training to the feature extraction process in the PBMEC context are explored.

## 1.7 Structure of Dissertation

This thesis is divided into three main parts with nine chapters in total. The first part, Chapters 1 and 2, provides an introduction to the research objectives and background of discriminative training methods. The second part focuses on the applications of MMI training to the HMM-based handwriting recognition systems, including the Thai handwriting recognition problem. This part consists of Chapters 3 to 6. The third part emphasises the applications of MCE training to the prototype-based classifiers (Chapters 7 and 8). The conclusions of this research are given in Chapter 9. Details of each chapter are summarised as follows:

**Chapter 2:** Discriminative Training

This chapter provides background knowledge of the statistical approach for pattern recognition which has led to the introduction of two popular discriminative training criteria: MMI and MCE.

**Chapter 3:** Hidden Markov Models and Discriminative Training

This chapter presents the concept of discriminative training, with the emphasis on MMI training, in the HMMs and the literature survey.

**Chapter 4:** MMI Training of Tied-Mixture Density Hidden Markov Models

This chapter investigates several aspects of MMI training in the TDHMM-based handwriting recognition systems, by using the on-line handwritten digit recognition task as a case study.

**Chapter 5:** Thai Character Recognition

In this chapter, an overview of the Thai character recognition problems is described. The reviews of the previous work are also presented.

**Chapter 6:** MMI Training of HMM-Based Thai Handwriting Recognition

In this chapter, the Thai handwriting recognition system based on the HMMs and MMI training is derived, and several techniques to improve its performance are studied.

**Chapter 7:** Discriminative Prototype-Based Recognition Systems

This chapter presents the concept of discriminative training in non-parametric classification, particularly MCE training in the prototype-based classifiers. The Prototype-Based Minimum Error Classifier (PBMEC) is explained in detail.

**Chapter 8:** Evaluation of Discriminative Prototype-Based Classifiers

In this chapter, the successful applications of PBMEC to the off-line unconstrained character recognition problems are described, and several key findings are revealed.

**Chapter 9:** Conclusions

Conclusions of this research and future research perspectives are given in this chapter.

# Chapter 2

# Discriminative Training

## 2.1 Introduction

The previous chapter (Section 1.2.1) explains the processes for solving the handwriting recognition problem, which is an instance of the pattern recognition problem. First, the handwritten characters are converted into feature vectors lying in feature space. The classification process then assigns the feature vector to one of the predefined categories. Generally speaking, it is a process that partitions the feature space into regions, one region for each category. These partitions, also known as the decision boundaries, are subsequently used to classify the unseen samples. An ideal classifier should provide a perfect separation so that wrong decisions never occur. However, in reality, this is hardly achieved. In practice, it is sufficient to have an optimal classifier that minimises the probability of error. This approach becomes a problem in statistical decision theory and thus is known as statistical pattern recognition. In order to understand the concept of finding the optimal decision boundaries, an introduction to the Bayes decision theory is essential as a fundamental statistical approach to the pattern recognition problem.

In this research, we focus on supervised learning in which the category of each sample is known and the decision boundaries can be derived from these observed samples. If all of the relevant probability distributions are known, the solution can be obtained simply. In practice, only some of these values are known and even the form of probability distributions has to be assumed. As a result, the goal turns into using the available training samples to estimate parameters of the underlying distributions. The most common method for parameter estimation is Maximum Likelihood (ML) estimation. The principal aspect of ML concentrates on *modelling* the data distribution for each category individually, and thus it is regarded as a non-discriminative training method. In contrast, discriminative training removes the requirement of having the correct underlying probability density. It aims to directly *separate* the categories through the design of effective decision boundaries in the feature space. Many discriminative training criteria have been proposed in the literature, mostly for speech recognition. Among them, two popular criteria are applied in this thesis: the Maximum Mutual Information (MMI) and

Minimum Classification Error (MCE) criteria. This chapter features a general overview of discriminative training criteria, without addressing their applications to specific classifiers. Their applications to Hidden Markov Models (HMMs) and prototype-based classifiers are extensively studied in Part II and Part III of this dissertation, respectively.

In this chapter, Bayes decision theory is briefly described in Section 2.2 followed by a discussion of Maximum Likelihood estimation in Section 2.3. Section 2.4 introduces different discriminative training criteria which include MMI, MCE and other discriminative approaches. The recent applications of discriminative training are reviewed in Section 2.5. The chapter closes with a summary in Section 2.6.

## 2.2 Bayes Decision Theory

The purpose of the classification problem is to determine the category to which the given pattern belongs. Bayes decision theory is a statistical method which formalises this procedure. We present the concept of this theory, based on the detailed discussions in [11, 12, 13], as background knowledge for discriminative training in later sections.

We begin with introducing the task of classifying an observed feature vector $\mathbf{x}$ into category $C_i$ which is one of $M$ categories, i.e. $C_i \in \{C_1, ..., C_M\}$. At this stage, we assume that all the relevant probability values are known and let $P(C_i)$ be the prior probability that the observed data belongs to each of the categories $C_i$, where $\sum_{i=1}^{M} P(C_i) = 1$. The *class-conditional* probability for $\mathbf{x}$, which is the probability density function for $\mathbf{x}$ given that the category is $C_i$, is also given and denoted as $P(\mathbf{x}|C_i)$. This information can be used to compute the posterior probability $P(C_i|\mathbf{x})$, which is the probability that the category is $C_i$ given the observation $\mathbf{x}$, by *Bayes Rule*:

$$P(C_i|\mathbf{x}) = \frac{P(\mathbf{x}|C_i)P(C_i)}{P(\mathbf{x})} \ , \tag{2.1}$$

where $P(\mathbf{x})$ is the density function for $\mathbf{x}$ irrespective of the category and is computed by:

$$P(\mathbf{x}) = \sum_{i=1}^{M} P(\mathbf{x}|C_i)P(C_i). \tag{2.2}$$

A decision rule is a function $\alpha(\mathbf{x})$ that tells us which category to decide for every possible observation $\mathbf{x}$. The action of making a decision that $\mathbf{x}$ belongs to category $C_i$ can be written as $\alpha(\mathbf{x}) = C_i$. The ultimate goal is to make a classification decision in such a way that the probability of error or misclassification is minimised. Given that the true category of $\mathbf{x}$ is $C_j$, the action of making a decision that $\mathbf{x}$ belongs to $C_i$ will incur the loss $\lambda(\alpha(\mathbf{x}) = C_i|C_j)$ (see later for a definition of $\lambda(\cdot)$). As a consequence, the expected loss associated with this action is:

$$R(\alpha(\mathbf{x}) = C_i|\mathbf{x}) = \sum_{j=1}^{M} \lambda(\alpha(\mathbf{x}) = C_i|C_j)\, P(C_j|\mathbf{x}). \tag{2.3}$$

The expected loss is also known as the conditional risk and the overall risk $R$ is given by:

$$R = \int R(\alpha(\mathbf{x})|\mathbf{x})\, P(\mathbf{x})\; d\mathbf{x}\ , \tag{2.4}$$

where $\alpha(\mathbf{x})$ can be any category, and the integral ranges over the entire feature space. In order to minimise the overall risk, the classification decision for every $\mathbf{x}$ has to be chosen so that the expected loss in Eq. 2.3 is as small as possible [11]. As a result, the *Bayes decision rule* for minimising the overall risk can be written in the general form as:

$$\text{decide } \mathbf{x} \in C_i \text{ if} \quad R(\alpha(\mathbf{x}) = C_i|\mathbf{x}) < R(\alpha(\mathbf{x}) = C_j|\mathbf{x}) \quad \text{for all } i \neq j. \tag{2.5}$$

By following this rule, it is promised that the classifier will provide the optimal partitioning of feature space among various categories.

In the classification problems where the ultimate goal is to minimise the probability of misclassification or error rate, the loss function $\lambda(\cdot)$ can be defined as:

$$\lambda(\alpha(\mathbf{x}) = C_i|C_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad \text{where } i, j = 1, .., M. \tag{2.6}$$

Using this loss function incurs no loss for making a correct decision for $\mathbf{x}$, and incurs a unit loss for any wrong decisions. Substituting Eq. 2.6 into 2.3 yields:

$$\begin{aligned} R(\alpha(\mathbf{x}) = C_i|\mathbf{x}) &= \sum_{j=1}^{M} \lambda(\alpha(\mathbf{x}) = C_i|C_j)\, P(C_j|\mathbf{x}) \\ &= \sum_{j \neq i} P(C_j|\mathbf{x}) \\ &= 1 - P(C_i|\mathbf{x}). \end{aligned} \tag{2.7}$$

Hence the resulting Bayes decision rule for the minimum error rate in Eq. 2.5 becomes:

$$\text{decide } \mathbf{x} \in C_i \text{ if} \quad P(C_i|\mathbf{x}) > P(C_j|\mathbf{x}) \quad \text{for all } i \neq j. \tag{2.8}$$

The classifiers that obey this decision rule are sometimes known as Maximum A Posteriori (MAP) classifiers. Obviously this can be seen as a formalisation of common-sense procedures, that is, the minimum probability of misclassification is obtained by assigning each new observation to the class for which the posterior probability is largest [12].

One particular way to represent the classifier is by considering that the classifier is composed of a set of *discriminant functions* $g_j(\mathbf{x})$, $j = 1, .., M$. The decision rule for the observed feature vector $\mathbf{x}$ is:

$$\text{decide } \mathbf{x} \in C_i \text{ if} \quad g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \text{for all } i \neq j. \tag{2.9}$$

In this way, the classification process can be viewed as a process that computes $M$ discriminant functions and chooses the category corresponding to the highest discriminant [11]. Obviously, there is more than one discriminant function that yields the same decision boundary. Any monotonically increasing function applied to the discriminant functions does not change the classification result. For example, from Eq. 2.8 and 2.9, the discriminant function for the Bayes classifier and its variations can be:

$$g_i(\mathbf{x}) = P(C_i|\mathbf{x}), \tag{2.10}$$

$$g_i(\mathbf{x}) = P(\mathbf{x}|C_i)P(C_i), \quad \text{or} \tag{2.11}$$

$$g_i(\mathbf{x}) = \log\, P(\mathbf{x}|C_i) + \log\, P(C_i). \tag{2.12}$$

As discussed earlier, the Bayes decision rule requires complete knowledge of all relevant probability values which is rarely acquired in practice. The prior probabilities $P(C_i)$ can be approximated with no difficulties. However, that is not the case for the class-conditional probabilities $P(\mathbf{x}|C_i)$. The target of designing a classifier eventually turns into using the available training samples to estimate $P(\mathbf{x}|C_i)$ as accurately as possible and to use the resulting estimates as if they were the true values. Section 2.3 explains the most common approach for this estimation, Maximum Likelihood, followed by the introduction to discriminative training approaches in Section 2.4.

## 2.3 Maximum Likelihood Estimation

The estimation of class-conditional probability density $p(\mathbf{x}|C_i)$ can be simplified by representing this density as a functional form, i.e. a *probability density function* (pdf), which consists of a number of adjustable parameters. This method is known as the *parametric* approach in which a specific form of the density function needs to be known, and the estimation of the probability density becomes a problem of estimating parameters of the underlying function. One of the most common and well known methods to solve this problem is *Maximum Likelihood* (ML) estimation. Detailed discussions of ML estimation for pattern recognition can be found in [11, 12]. Here, we describe its essential concepts and limitations which lead to the introduction of discriminative training in Section 2.4.

First, we need to make an assumption that the correct functional form of $P(\mathbf{x}|C_i)$ is known. Therefore, $P(\mathbf{x}|C_i)$ can be uniquely determined by the values of its parameter vector $\boldsymbol{\theta}_i$ where the vector contains all parameters of the chosen functional form. The $P(\mathbf{x}|C_i)$ can be written as $P_{\boldsymbol{\theta}_i}(\mathbf{x}|C_i)$ to explicitly show that its value is dependent on $\boldsymbol{\theta}_i$. In the classification problem, we assign one such function to each of the categories. Suppose there is a set of $R$ training samples, $\boldsymbol{\mathcal{O}} = \{\boldsymbol{\mathcal{O}}_1, ..., \boldsymbol{\mathcal{O}}_r, ..., \boldsymbol{\mathcal{O}}_R\}$. Each sample $\boldsymbol{\mathcal{O}}_r$ is to be classified into one of the $M$ categories $\{C_1, ..., C_M\}$, where $\boldsymbol{\mathcal{O}}_r$ is the $r$-th training sample. We also have a knowledge of its correct category $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^r, ..., \mathcal{C}^R\}$, where $\mathcal{C}^r$ is the correct category of $\boldsymbol{\mathcal{O}}_r$ and $\mathcal{C}^r \in \{C_1, ..., C_M\}$. The target is to acquire a good estimate of $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M\}$ using the available training samples $\boldsymbol{\mathcal{O}}$.

If the samples are drawn independently from the distribution $P(\mathbf{x}|C_i)$, then the joint probability density of the whole training set $\boldsymbol{\mathcal{O}}$ given the corresponding category set $\mathcal{C}$ is given by:

$$P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}|\,\mathcal{C}) = \prod_{r=1}^{R} P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r) \;\equiv\; \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}), \tag{2.13}$$

where $\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}})$ is referred to as the *likelihood* of $\boldsymbol{\theta}$ for the given $\boldsymbol{\mathcal{O}}$. Since the likelihood is a function of $\boldsymbol{\theta}$ for the fixed sample set $\boldsymbol{\mathcal{O}}$, it can be alternatively written as $\mathcal{L}(\boldsymbol{\theta})$ [12]. By its definition, maximum likelihood estimation of $\boldsymbol{\theta}$ is the process to find the value $\hat{\boldsymbol{\theta}}$ that maximises $\mathcal{L}(\boldsymbol{\theta})$. Intuitively, the process of ML estimation is to find the parameters that give rise to the observed data, i.e. it is the process of trying to model the data. Although the models for all categories need to be found, the above equation suggests that the estimate of each category can be computed independently by using only the samples that belong to that category. In particular, the estimation of $\boldsymbol{\theta}_i$ only takes into account the samples that belong to category $C_i$.

From the above equation, it is also obvious that applying any monotonically increasing function to the likelihood does not affect the solution $\hat{\boldsymbol{\theta}}$. In practice, it is more convenient to maximise the log-likelihood than the likelihood itself, i.e.

$$\log\,\mathcal{L}(\boldsymbol{\theta}) = \log\,P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}|\,\mathcal{C}) = \sum_{r=1}^{R} \log\,P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r) \;\equiv\; f_{ML}(\boldsymbol{\theta}), \tag{2.14}$$

because the products of probabilities are converted into their summations and the estimate $\hat{\boldsymbol{\theta}}$ that maximises the log-likelihood also maximises the likelihood. Eq. 2.14 is also known as the *ML objective function* or $f_{ML}$, which is maximised during the training process. After parameter estimation has been done, the log-likelihood can be used as a discriminant function to directly determine a decision boundary, based on the Bayes decision rule (Eq. 2.12), given that the prior probability $P(C_i)$ is provided.

The difficulty of maximising log-likelihoods depends upon the functional form of $P(\mathbf{x}|C_i)$. For example, assuming $P(\mathbf{x}|C_i)$ is a Gaussian distribution (normal density), its model parameters $\boldsymbol{\theta}_i$ can be viewed as $\boldsymbol{\theta}_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$, where $\boldsymbol{\mu}_i$ is the mean vector and $\boldsymbol{\Sigma}_i$ is the covariance matrix, since the normal density is a function that can be uniquely identified by its mean and covariances. In this case, the pdf is a differentiable function. Therefore, by setting the differential of log-likelihood to zero, the standard method of differential calculus can be applied to find $\hat{\boldsymbol{\theta}}_i$. However, for most choices of density function, finding $\hat{\boldsymbol{\theta}}_i$ is almost analytically intractable using the standard method. The optimisation has to be done using iterative techniques. The Expectation Maximisation (EM) algorithm [14] is one of the popular and powerful techniques for this purpose. The work in [15] provides a detailed description of the EM algorithm. We describe the EM algorithm in more detail in Chapter 4, where it is particulary applied to ML training for the HMMs. Maximum likelihood estimation is considered an efficient technique for classifier design and has shown its popularity in many applications, including the estimation of HMM parameters. The most compelling factors for the success of ML training in the HMMs are the existence of the powerful iterative method derived from the EM algorithm to estimate the parameters and the fact that this algorithm is mathematically guaranteed to converge (see Chapter 4).

From our discussion, the success of ML estimation is, so far, based on the assumption that the correct functional form of probability density is known. Nádas [16] has shown that if the samples have the same distribution as the assumed distribution and the number of samples is large enough, the maximum likelihood estimate will be the best representation of the true parameters. For example, the ML estimation of the mean and variances of a Gaussian density function will be the best estimate of the true mean and variances, if the samples are normally distributed as assumed. Although this may be true, neither condition can be achieved in practice. The patterns in the actual problems, such as speech or handwriting samples, are so complex that the modelling of its correct distribution, even if possible, requires a large number of parameters, and the number of training samples is always limited. Hence the optimality of ML estimation is based on assumptions which are not valid in practice. A simple demonstration of this problem can be found in [17].

Because ML estimation focuses on modelling the distribution of samples in each category individually, it is regarded as a non-discriminative training method. Having considered the Bayes decision rule carefully, it can be seen from Eq. 2.8 that we do not need to represent the posterior probability $P(C_i|\mathbf{x})$ explicitly, as long as we can identify which category $C_i$ provides the highest $P(C_i|\mathbf{x})$.

## 2.4 Discriminative Training Criteria

In contrast to ML training, discriminative training removes the requirement of knowing the correct functional form of the sample distributions. Rather than trying to model the data correctly, discriminative training aims to separate the categories by taking into account the samples of other competitive categories. The training criteria can be explained by considering their objective functions in which the settings of parameters in these functions reflect the performance of the classifiers. Discriminative training procedures try to find the parameters that optimise (either maximise or minimise) these objective functions. We discuss in detail two important discriminative training criteria, namely MMI and MCE criterion, as followed.

### 2.4.1 Maximum Mutual Information Criterion

The use of Maximum Mutual Information (MMI) applied to pattern recognition problems, especially speech recognition, was introduced by Bahl *et al.* [18] as an alternative to ML estimation. MMI estimation aims at finding the parameter set which maximises the mutual information between the samples and their correct categories. MMI estimation derives from the basic concept of mutual information known in the information theory. Generally speaking, the mutual information between two random variables measures the amount of information that one conveys about the other. Let $X$ and $Y$ be random variables and $x$ and $y$ be their respective outcomes. From [19, 20], the mutual information between $X$ and $Y$, $I(X;Y)$, is defined as:

$$I(X;Y) = H(X) - H(X|Y) \ , \tag{2.15}$$

where $H(X)$ is an entropy of $X$ and $H(X|Y)$ is a conditional entropy of $X$ given $Y$, and are defined by:

$$H(X) = -\sum_{x \in X} P(x) \log\, P(x) \ , \tag{2.16}$$

$$H(X|Y) = -\sum_{x \in X} \sum_{y \in Y} P(x,y) \log\, P(x|y). \tag{2.17}$$

In other words, the mutual information measures the reduction in uncertainty about $X$ that results from learning about $Y$. From the above equations, $I(X;Y)$ can be derived as follows:

$$\begin{aligned} I(X;Y) &= -\sum_{x \in X} P(x) \log\, P(x) - \left( -\sum_{x \in X} \sum_{y \in Y} P(x,y) \log\, P(x|y) \right) \\ &= -\sum_{x \in X} \sum_{y \in Y} P(x,y) \log\, P(x) + \sum_{x \in X} \sum_{y \in Y} P(x,y) \log\, P(x|y) \\ &= \sum_{x \in X} \sum_{y \in Y} P(x,y) \log\, \frac{P(x|y)}{P(x)}. \end{aligned} \tag{2.18}$$

In this way, it can be seen that $I(X;Y)$ is the average, taken over the $X$, $Y$ sample space, of the random variable:

$$I(x;y) = \log\, \frac{p(x|y)}{p(x)}. \tag{2.19}$$

The MMI criterion is then derived from this point. Given a set of $R$ training samples $\boldsymbol{\mathcal{O}} = \{\boldsymbol{\mathcal{O}}_1, ..., \boldsymbol{\mathcal{O}}_r, ..., \boldsymbol{\mathcal{O}}_R\}$ and its corresponding category set $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^r, ..., \mathcal{C}^R\}$, where $\boldsymbol{\mathcal{O}}_r$ is the $r$-th training sample and $\mathcal{C}^r$ is the correct category of $\boldsymbol{\mathcal{O}}_r$, the mutual information $I(\boldsymbol{\mathcal{O}}_r, \mathcal{C}^r)$ between the training sample $\boldsymbol{\mathcal{O}}_r$ and its corresponding category $\mathcal{C}^r$ can be derived from Eq. 2.19 as follows:

$$\begin{aligned} I(\boldsymbol{\mathcal{O}}_r; \mathcal{C}^r) &= \log\, \frac{P(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r)}{P(\boldsymbol{\mathcal{O}}_r)} \\ &= \log\, P(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r) - \log\, P(\boldsymbol{\mathcal{O}}_r) \\ &= \log\, P(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r) - \log \sum_{i=1}^{M} P(\boldsymbol{\mathcal{O}}_r|C_i)P(C_i). \end{aligned} \tag{2.20}$$

Given the functional form of the conditional probabilities whose values can be identified by their model parameters $\boldsymbol{\theta}$, the mutual information and the conditional probabilities can be written with the subscription $\boldsymbol{\theta}$. To apply MMI estimation to the pattern recognition problem, the idea is to estimate the model parameters $\boldsymbol{\theta}$ that maximise the mutual information between the training samples $\boldsymbol{\mathcal{O}}$ and their correct categories $\mathcal{C}$. This is equivalent to maximising the *MMI objective function*, $f_{MMI}$, i.e. the average of the mutual information across all training samples which is defined as:

$$f_{MMI}(\boldsymbol{\theta}) = \frac{1}{R} \sum_{r=1}^{R} \left( \log\, P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r) - \log \sum_{i=1}^{M} P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|C_i)P(C_i) \right). \tag{2.21}$$

The maximisation of $f_{MMI}$ aims at maximising the *difference* between $P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r)$ and the density probability $P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r) = \sum_{i=1}^{M} P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|C_i)P(C_i)$, whereas ML estimation (Eq. 2.14) attempts to maximise only the conditional probabilities $P_{\boldsymbol{\theta}}(\boldsymbol{\mathcal{O}}_r|\mathcal{C}^r)$. In other words, MMI training not only tries to maximise the conditional probability of the training samples given their correct categories, it also tries to minimise the corresponding conditional probabilities of all competitive classes. The parameter set $\boldsymbol{\theta}$ is chosen to improve the discrimination among the training samples. Thus, the MMI estimation is regarded as a discriminative training method. It focuses on the *separation* of the classes rather than trying to explicitly model the posterior probabilities as in ML training. For this reason, MMI estimation has an advantage over ML estimation in that it does not require the assumption that the functional form of the probability distribution is correctly specified [21]. Nádas *et al.* [22] has demonstrated that, in some circumstances, the use of MMI training with an incorrect functional form of probability converges to an optimal result given sufficient training samples, whereas ML estimation does not.

It should be noted that maximising the MMI objective function requires more computation than maximising the likelihood. For each training sample, we need to calculate the class-conditional probabilities for all categories, not only the correct class. Part II of this dissertation is devoted to the practical applications of MMI training to the HMM-based classifiers. Section 3.5 explains the optimisation methods for the MMI criterion while its applications to the handwriting recognition problems are demonstrated in Chapters 4 and 6.

### 2.4.2 Minimum Classification Error Criterion

It can be seen from Section 2.2 that the natural target of designing the classifiers should be to minimise the classification errors. Despite the success of MMI training in many applications, there is no direct relationship between optimising the MMI criterion and minimising the probability of classification error. In this section, we examine an alternative to the discriminative training criteria which focuses directly on minimising the empirical classification error, namely the Minimum Classification Error (MCE) criterion. The concept of MCE training is based on the early work of Amari [23]. Researchers in the speech recognition community became aware of MCE training when Katagiri *et al.* [24, 25] proposed a comprehensive theoretical framework for the optimisation of the MCE objective function. A detailed description of MCE training can be found in [17, 26]. Its basic ideas are presented below.

The essential aspect of MCE training is to define a loss function as a function of trainable parameters of the classifier that is proportional to the classification error. Given a set of $R$ training samples $\boldsymbol{\mathcal{O}} = \{\boldsymbol{\mathcal{O}}_1, ..., \boldsymbol{\mathcal{O}}_r, ..., \boldsymbol{\mathcal{O}}_R\}$, the correct category of $\boldsymbol{\mathcal{O}}_r$ is known as $C_k \in \{C_1, ..., C_M\}$, where $M$ is the total number of categories. Note that the notation used in this section is slightly different from the one in Sections 2.3 and 2.4.1. The training process tries to estimate the model parameter $\boldsymbol{\theta}$ that minimises the *MCE objective function* defined as:

$$f_{MCE}(\boldsymbol{\theta}) = \frac{1}{R}\sum_{r=1}^{R} \ell_k(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta}), \tag{2.22}$$

where $\ell_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})$ is the loss occurred for classifying the sample $\boldsymbol{\mathcal{O}}_r$ given its correct category $C_k$. The loss function in Eq. 2.6 can be rewritten here as:

$$\ell_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta}) = \begin{cases} 0 & \text{if } \boldsymbol{\mathcal{O}}_r \text{ is correctly classified as } C_k \\ 1 & \text{if } \boldsymbol{\mathcal{O}}_r \text{ is misclassified as } C_j\text{, where } j \neq k. \end{cases} \tag{2.23}$$

This is the ideal loss function that should be applied to the MCE objective function since it directly reflects the classification error. However, in practice, the estimation for $\boldsymbol{\theta}$ that minimises the MCE objective function is performed using gradient-based optimisation approaches which require a loss function which can be differentiated. At least a first-order differentiable function is needed. Therefore, this ideal loss is not applicable to MCE, as is not differentiable. The MCE approach approximates the ideal loss by a continuous, differentiable loss function which is more suitable to gradient-based optimisation. The practical loss function is defined as a function of the *misclassification measure*, $d_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})$:

$$\ell_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta}) = \ell(d_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})), \tag{2.24}$$

which maps the misclassification measure to a value between 0 and 1. Obviously there are a lot of choices for the loss function $\ell(d)$. One function is the sigmoid function given by:

$$\ell(d) = \frac{1}{1+e^{-\alpha d}} \tag{2.25}$$

with a large positive $\alpha$. This loss limits are 0 and 1 for $d \to -\infty$ and $d \to \infty$, respectively. This approximates the ideal loss well and is differentiable for all values of $d$. An alternative approximation is to use a simple piece-wise linear loss defined as:

$$\ell(d) = \begin{cases} 0 & \text{if } d < Q1 \\ (d-Q1)/(Q2-Q1) & \text{if } Q1 \leq d < Q2 \\ 1 & \text{if } d \geq Q2. \end{cases} \tag{2.26}$$

Prior to giving the definition of a misclassification measure $d_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})$, let us first consider the definition of a discriminant function mentioned in Section 2.2. The discriminant function $g_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})$ reflects the extent to which a sample $\boldsymbol{\mathcal{O}}_r$ belongs to the category $C_k$. The classification decision of $\boldsymbol{\mathcal{O}}_r$ is made by choosing the category that yields the highest discriminant value (from Eq. 2.9), that is:

$$\text{decide } \boldsymbol{\mathcal{O}}_r \in C_k \text{ if} \quad g_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta}) > g_j(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta}) \quad \text{for all } j \neq k. \tag{2.27}$$

From this point, the misclassification measure will be defined as a function of the discriminant function. Similar to the loss function, the discriminant function is also required to be continuous and differentiable while the choice of the classifier determines the form of this function. The misclassification measure of $\boldsymbol{\mathcal{O}}_r$ is defined as a comparison among the discriminant function for the correct category $C_k$, $g_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})$, and the discriminant functions of all other categories, $g_j(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})$ where $j \neq k$. It maps the classification decision into a scalar value. Corresponding to the loss function we define, a positive sign of the misclassification measure implies an incorrect decision and a negative sign implies a correct decision. Amari [23] proposed the following misclassification measure:

$$d_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta}) = \frac{1}{N_k} \sum_{j \in \mathcal{A}_k} (g_j(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta}) - g_k(\boldsymbol{\mathcal{O}}_r;\boldsymbol{\theta})), \tag{2.28}$$

where the set $\mathcal{A}_k$ is composed of the indexes of the $N_k$ categories whose discriminant function values are higher than that of the correct category $C_k$, i.e. $\mathcal{A}_k = \{j | g_j(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta}) > g_k(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta})\}$. When $\mathcal{A}_k$ is empty, $d_k(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta})$ is set to zero. Thus, when the correct classification occurs, this misclassification measure does not give much information of the degree of difference between the correct category and the incorrect categories. A more informative measure is desirable in most cases. Moreover, the above misclassification measure is discontinuous given that the set $\mathcal{A}_k$ is not fixed since it is changed according to $\boldsymbol{\theta}$. It is therefore not suitable to be applied to MCE.

Twenty-four years later, Katagiri *et al.* [24] introduced the following misclassification measure to overcome these limitations:

$$d_k(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta}) = -g_k(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta}) + \left[ \frac{1}{M-1} \sum_{\substack{j=1 \\ j \neq k}}^{M} g_j(\boldsymbol{\mathcal{O}}_r; \boldsymbol{\theta})^{\eta} \right]^{\frac{1}{\eta}}. \tag{2.29}$$

This misclassification measure has been widely used in MCE training. It is continuous over the model parameter $\boldsymbol{\theta}$ and, at the same time, provides an indication of how good the classification is. This measure can be viewed as a difference between the discriminant value of the correct category and the average of the discriminant values of competing categories. The parameter $\eta$ is a positive constant, which controls the degree of contribution of competing categories. For example, when $\eta$ is large, the second term of Eq. 2.29 is approximated to the highest discriminant value among the incorrect categories, and the resulting misclassification measure resembles the direct comparison between the discriminant function of the correct category and its best competitive category.

The formalisation of the MCE criterion described above shows that the MCE training process attempts to minimise the classification error directly rather than trying to model the posterior probability. It allows, but does not require, the use of explicit probabilistic models which can overcome the limitation of using an incorrect probabilistic model. Comprehensive studies of the link between MCE and the Bayes decision theory are given in [17, 27]. In practice, the MCE objective function, $f_{MCE}$, is minimised by the gradient-based optimisation techniques such as Generalised Probabilistic Descent (GPD) [28] or the approximated Newton-based optimisation known as QuickProp [29]. The term MCE/GPD is often referred to in speech recognition research where GPD is used to train system parameters. A detailed optimisation process of MCE is presented in Chapter 7, together with its applications to the prototype-based classifiers for handwriting recognition in Chapter 8.

### 2.4.3 Other Discriminative Training Criteria

Many variations of discriminative training criteria have been reported in the literature. The objective function, which is to be either maximised or minimised, is the essential factor for the success of each discriminative training criterion. Most of the recently proposed objective functions are specially designed for speech recognition tasks. For example, Povey [10] introduced Minimum Phone Error (MPE) and Minimum Word Error (MWE) which are more directly related to *word error rate*, the scoring criterion generally used in continuous speech recognition. A modification of MMI objective function called non-linear or Sigmoid-MMI (SMMI) was proposed by Valtchev [30], with the attempt to increase more discriminative ability among competitive categories. Generally, these discriminative training criteria, including MMI and MCE, are applied to the non-discriminative classifiers such as the HMMs and Gaussian Mixture Model (GMM) classifiers [12]. On the other hand, neural networks classifier is considered discriminative by its nature since parameter estimation for all categories is performed simultaneously using all available training samples. The criterion typically used to train neural networks is the Mean Squared Error (MSE), although in some cases MMI and MCE can be applied.

## 2.5 Applications of Discriminative Training

The use of discriminative objective functions incorporates an explicit comparison among competitive categories during the training process. It focuses on separating the categories through the adjustments of system parameters, which remove the requirement that the correct form of probability density needs to be known. Although discriminative training involves a lot more computation than ML training, a number of powerful optimisation algorithms have been proposed recently and result in an increasing number of discriminative training applications. Discriminative training has demonstrated potential improvements in recognition performance compared to ML training in many applications. However, most research has been done in speech recognition with only a few reports addressing discriminative training usage in other applications, including character recognition. Following is a brief summary for these applications.

### 2.5.1 Applications in Speech Recognition

In speech recognition, the state of the art of training criteria had been ML estimation of the HMM parameters. However, awareness of the discriminative training criteria has increased significantly over the past decade. Discriminative training is introduced in an attempt to overcome the limitation of ML in which the model assumptions need to be correct and sufficient training samples be available. An improvement in the recognition performance has been demonstrated in the literature when discriminative training is applied. The applications of the two important discriminative criteria: MMI and MCE, for speech recognition can be summarised as following.

The work by Bahl *et al.* [18] was among the first experiments with the use of MMI training for an HMM-based isolated word recognition system. A relative improvement in the word error rate of 18% was reported, in comparison to ML training. Shortly after, Brown [21] reported the gain in performance by using MMI training in different systems. Early work of MMI training relied on the gradient-based approaches for the optimisation of HMM parameters. An extension of the EM algorithm for the optimisation of the MMI objective function was later introduced in [31]. Normandin *et al.* [32] subsequently applied the modified version of this algorithm and provided significant evidence in favour of MMI training. Among the recent important results, the work in [33, 34] applied MMI training to a large-scale speech recognition system and led to significant reductions in error rate compared to the results from ML training. Apart from its success in the HMM-based systems, the MMI criterion was also applied to the neural networks [35] and Gaussian Mixture Models classifiers [36], mainly for discriminative feature transformation applications.

Likewise, MCE training has been extensively applied to speech recognition deploying various classifier structures. Variations of loss functions and misclassification measures have been proposed. Further detailed explanations can be seen in [17]. MCE training has been frequently referred to as MCE/GPD since its powerful optimisation algorithm called Generalised Probabilistic Descent (GPD) was introduced by Katagiri *et al.* [24, 25]. The applications of MCE in speech recognition using HMMs were reported in [17, 37] in which the recognition results outperformed the non-discriminative ML estimation. Another successful application of MCE training was demonstrated in [25] where it was applied to a neural networks classifier. McDermott and Katagiri [38] implemented MCE/GPD within the prototype-based speech recognition framework. Besides, the exhaustive study by Biem [27] was among the significant contributions of discriminative feature transformation using MCE.

### 2.5.2 Applications in Character Recognition

Although discriminative training has proved its success in speech recognition applications for a decade, there has been little awareness of its effectiveness in other application areas including the area of character recognition. A number of reports regarding the applications of discriminative training in character recognition have just emerged over the past few years. Among the early work, Bengio *et al.* [39] employed the MMI criterion for the joint training of neural networks and for post-processing, with an attempt to minimise word-level errors. The experimental results have shown considerable reduction in error rates on the testing set. Recently, the application of a MCE-trained Gaussian Mixture Model (GMM) for isolated handwritten character recognition in [40] has demonstrated better performance compared to the ML-based method. The performance of MCE training applied to the neural networks classifier for word level discrimination is shown in [41]. Of the significant results, Biem successfully trained the HMM classifiers for on-line character [42] and word recognition [43] using the MCE criterion and indicated improvements in recognition results. Another important result of discriminative training includes the work by Huo *et al.* [44] where the MCE criterion was concurrently applied to the model training and discriminative feature extraction process in the prototype-based classifier. It has reported very high recognition results when applied to printed Chinese character recognition. Being inspired by the same work in [38], the aforementioned work [44] (published in 2001) and this research coincidentally share a similar classifier structure despite the significant difference in the defined discriminant functions.

## 2.6 Summary

This chapter provides background knowledge of the statistical approach for pattern recognition which has led to the introduction of discriminative training. In particular, the Bayes decision theory serves as a formalisation of the classification process in order to achieve the optimal decision boundaries. As a result, the goal of pattern recognition turns into finding model parameters that minimise the error rate by using the available training samples. Maximum Likelihood (ML) estimation, which is the most common method for parameter estimation, is described. ML training is considered as a non-discriminative training approach because it aims at *modelling* the data distribution rather than directly *separating* the categories. ML estimation relies on the assumption that the correct functional form of probability density is known, which is not true in practice. Discriminative training techniques are introduced as alternatives to ML estimation in order to overcome its limitations. Discriminative training attempts to adjust the model parameters to produce better decision boundaries. Two important discriminative training approaches, the Maximum Mutual Information (MMI) and Minimum Classification Error (MCE) criteria, have been discussed. MMI training aims at maximising the mutual information between the samples and their correct categories and is more discriminative than ML estimation. In contrast, MCE focuses directly on minimising the classification errors. Although both methods require more computation than ML estimation, and have indirect relationship to the Bayes decision theory, they have demonstrated success in many applications with potential improvements in the recognition performance compared to ML training.

There has been only a little evidence in applying discriminative training to character recognition problems, despite its accomplishment in many applications of speech recognition over the past decade. This has motivated this research. Part II and Part III of the dissertation are devoted to the applications of discriminative training to handwriting recognition. We particularly address the use of discriminative training in the HMMs and the prototype-based classifiers.

# Part II

# Discriminative Training in Hidden Markov Models

# Chapter 3

# Hidden Markov Models and Discriminative Training

## 3.1 Introduction

The *parametric* approach to pattern recognition provides a simplified method for the estimation of the class-conditional probability densities, as required by the Bayes classifiers. In this case, an assumption for the correct functional form of the pdf has to be made, thereby converting the classifier-designing problem into estimating parameters of the underlying function. Among several techniques of the parametric approach, the Hidden Markov Model (HMM) classifier is one of the most widely used techniques and has become standard in the field of speech recognition. Recently, the HMMs have also been extensively deployed in handwriting recognition applications, both on-line and off-line, as seen from a large number of research reports published over the past decade [3]. The training process in the HMMs is usually performed by Maximum Likelihood (ML) estimation through a powerful optimisation algorithm known as the Baum-Welch algorithm. In this dissertation, discriminative training in the HMMs is introduced as an alternative approach to ML estimation and has demonstrated an improvement in the recognition performance when compared to ML training. Substantial effort has been devoted to the applications of discriminative training in the HMMs to speech recognition problems. However, discriminative training is an ongoing research area, and many of its aspects have not been explored. Much effort is also required to evaluate its performance in the handwriting recognition tasks.

The main structure of Part II (Chapters 3 to 6) is concerned with discriminative training in the HMMs and its applications to the handwriting recognition problems. This chapter presents the concept of discriminative training with the emphasis on Maximum Mutual Information (MMI) training in the HMMs as background knowledge for our original studies in Chapters 4 and 6. The introduction to the HMM-based classifier is given in Section 3.2. Section 3.3 explains ML training of HMM parameters, which is based on the Expectation-Maximisation (EM) algorithm. Section 3.4 provides a comprehensive review of the applications of the HMMs to handwriting recognition. The optimisation of the HMM parameters based on the MMI criterion is extensively discussed in Section 3.5. Recent applications of discriminative training in the HMMs are reviewed in Section 3.6. A summary of the chapter is given in Section 3.7.

## 3.2 Hidden Markov Model Background

The Hidden Markov Model (HMM) is a mathematical model of stochastic processes which generates random sequences of symbols as determined by certain probabilities. It has been used to model time-varying (or nonstationary) signals whose statistical properties change over time. The HMM relies on a reasonable approximation which represents a time-varying signal as a sequence of time-invariant (or stationary) signal segments given that, presumably, the properties of signal do not vary significantly within a short period of time. The underlying concept of the HMM is similar to that of the Markov model, except that in the HMM we can only observe the sequence of outputs generated by the model, but the information of the state sequence is not accessible, hence its name–*Hidden* Markov Model. The output sequence generated by the HMM is often referred to as the *observation* sequence. Using HMMs as a classifier replaces the problem of estimating the class-conditional observation densities with a simpler problem of estimating the HMM parameters from the available samples. The HMMs have demonstrated their success in the field of speech recognition and, recently, in handwriting recognition. Section 3.4 summarises the applications of the HMMs to handwriting recognition problems.

Prior to introducing discriminative training in the HMMs, it is necessary to understand the basic principles of the HMMs and their conventional training process. Although the HMM theory has been extensively explained in the literature [45], consistent notation throughout the dissertation is required for better understanding and completeness. Following sections present a brief introduction to the theory behind the HMMs, including some issues regarding its practical usage.

### 3.2.1 Representation of the HMM

An effective way to explain the HMM is to view it as a finite state machine which generates the output segment $\mathbf{o}_t$ at each time step $t$ while being at state $q_t$ and changes to the next state $q_{t+1}$ once every time step $t+1$. This process continues until the model reaches the maximum time step $T$. As a result, a sequence of output segments or the observation sequence $\mathbf{O}^T$ is produced, where $\mathbf{O}^T = \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t, ..., \mathbf{o}_T\}$. Every state in the HMM can be reached from every other state of the model, including itself, thereby resulting a fully-connected (or *ergodic*) model topology. However, the *left-right* model topology is normally used since it is more capable of modelling a time-varying signal. Its important property is that, as the time step increases, the state can only be moved to either the same state or the state that has an increasing index, i.e. proceeding from left to right. The $N$-state left-right model is forced to start at state 1 and end in state $N$ at the time step $T$. The left-right model topology is widely used in the HMMs for speech and handwriting

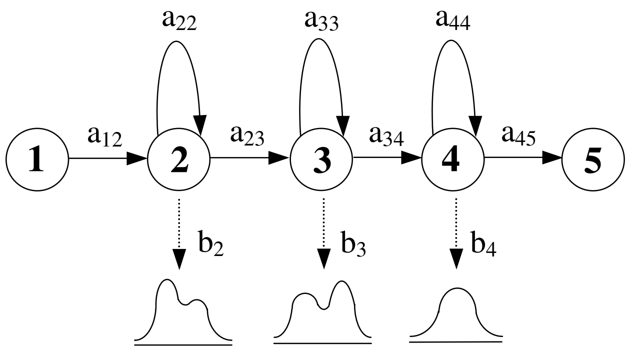


**Figure 3.1:** An HMM having left-right topology often used in speech and handwriting recognition.

recognition and so is in this work. Additional constraints are usually imposed to limit large changes of state index occurred in each transition. In our models, no transitions of more than one consecutive state are allowed.

Figure 3.1 shows an example of the 5-state left-right HMM. This model is composed of five states, $s_1$ to $s_5$. It also demonstrates two important properties of the HMM. First, the transition from one state to another state in the HMM is determined by a probabilistic transition function. This function is described by an $N \times N$ matrix of discrete probabilities $a_{ij}$, known as the state transition probability matrix $\mathbf{A}$. The first-order HMM has a property similar to the first-order Markov model in a sense that the probability of being in the current state only depends on the previous state and not the ones before it [45], or alternatively the following condition is satisfied:

$$P(q_t = s_j | q_{t-1} = s_i, q_{t-2} = s_k, ...) = P(q_t = s_j | q_{t-1} = s_i). \tag{3.1}$$

Each element $a_{ij}$ in $\mathbf{A}$ represents the probability of being in state $s_j$, given that the transition is made from the previous state $s_i$. This is called the *transition probability* and is defined as:

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i) \quad \text{where } 1 \leq i, j \leq N. \tag{3.2}$$

Also, note that the sum of all transition probabilities that leaves any state is unity, i.e. $\sum_{j=1}^{N} a_{ij} = 1$, $\forall i$. Although any order of state transition probabilities can be used in the HMM, the first-order HMM is commonly deployed in most pattern recognition problems.

Second, as opposed to the Markov model, it is shown that every state in the HMM is associated with a probability density function for the production of particular observations. The probability of generating the observation $\mathbf{o}_t$ while being in state $s_j$ is defined as

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t | q_t = s_j) = b_{q_t}(\mathbf{o}_t) \quad \text{where } 1 \leq j \leq N. \tag{3.3}$$

In practice, the *non-emitting* states, the states that do not generate the outputs, are often placed in the model as the start and end states as shown in Figure 3.1. These states assume no increments of the time step $t$ while entering the state and are useful for the connection of several HMMs. An $1 \times N$ matrix $\mathbf{B}$ contains a collection of all states' pdfs, $b_j$. The HMM can be represented by its model parameters, $\lambda$, which are composed of the state transition probability matrix $\mathbf{A}$ and the output probability matrix $\mathbf{B}$, i.e. $\lambda = \{\mathbf{A}, \mathbf{B}\}$.

Apart from the HMM structure described in this section, some researchers prefer dealing with another structure in which outputs are generated by transitions into states and not by states. That is to say, the output $\mathbf{o}_t$ is produced at each time step $t$ while the transition from state $q_{t-1}$ to $q_t$ is being made. The advantage of this HMM structure is concerned with its flexibility to introduce the possibility of *null* transitions that change state but result in no output. This is useful for modelling an observation sequence that has fewer output segments than the number of HMM states. However, as described in [46], these two HMM structures are entirely equivalent and interchangeable, and the formulae used to compute probabilities in one structure can also be adapted to another structure. The formulations of the HMM in the following sections are based on the structure described in this section in which outputs are produced by states.

### 3.2.2 Output Probability

This section presents the properties of the output probability functions, $b_j$. There are two main variations of the state output pdf: the *discrete* and *continuous* distributions. On the one hand, the *discrete* density provides an efficient method for modelling the data which are naturally symbolic. Each observation vector of a continuous signal is quantised to give a discrete symbol by using Vector Quantisation (VQ) [47] and a predefined number of prototype vectors (or so-called *codebook*). In this instance, $b_j$ is a vector whose dimension equals to the codebook dimension, and each element corresponds to the probability of generating the associated prototype vector. Although the use of the discrete density HMM is computationally inexpensive, it is not by nature suitable for continuous signals.

On the other hand, the *continuous* output density is generally preferable for modelling the continuous signals. In this case, $b_j$ is a parametric distribution of a predetermined function. The most commonly used distribution is the Gaussian density function. The mixtures of Gaussian densities are often employed for a more accurate approximation. In this case, the output density at state $s_j$ is calculated by:

$$\begin{aligned} b_j(\mathbf{o}_t) &= \sum_{m=1}^{M} c_{jm}\, b_{jm}(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm}\, \mathcal{N}(\mathbf{o}_t, \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \\ &= \sum_{m=1}^{M} c_{jm} \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_{jm}|}} \exp\left(-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{jm})' \, \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{o}_t - \boldsymbol{\mu}_{jm})\right), \end{aligned} \tag{3.4}$$

where $D$ is the dimension of the observation vector $\mathbf{o}_t$, and $M$ is the number of mixtures. The *mixture coefficient*, $c_{jm}$, is a parameter that controls the contribution of the $m$-th Gaussian mixture, and $\sum_{m=1}^{M} c_{jm} = 1$, while $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$ serve as the $m$-th mixture's mean vector and covariance matrix, respectively. The HMM that employs this output density is known as the continuous density HMM (CDHMM). In practice, it is assumed that the elements in the observation vector are uncorrelated. This yields the diagonal covariance matrix whose off-diagonal elements are set to zero.

An alternative output distribution, known as the semi-continuous density or tied-mixture density HMM (TDHMM) where a pool of Gaussian densities is shared among several states and models, is powerful and capable of reducing a number of model parameters significantly. Both CDHMMs and TDHMMs are used in this research (see Chapter 4 for more details).

### 3.2.3 Probability Computation

For isolated character recognition, each character category is modelled by one HMM. The models for all character classes are used to classify the unknown samples $\mathbf{O}^T$. According to the Bayes decision theory described in Section 2.2, the model that produces the highest discriminant function is chosen as the recognised category of $\mathbf{O}^T$. Since the model parameters $\lambda$, representing class $C$, has already been defined, a discriminant function for each HMM can be represented by the likelihood $P(\mathbf{O}^T|\lambda)$, the probability that the observation sequence $\mathbf{O}^T$ is generated by the model $\lambda$, multiplied by the prior probability $P(C)$. Given $\mathbf{O}^T = \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T\}$, the likelihood probability can be calculated by:

$$P(\mathbf{O}^T|\lambda) = \sum_{Q} P(\mathbf{O}^T, Q|\lambda), \tag{3.5}$$

where $P(\mathbf{O}^T, Q|\lambda)$ denotes the the probability that the model generates the observation sequence $\mathbf{O}^T$ and has the corresponding state sequence $Q$. Note that we can only observe $\mathbf{O}^T$ but not the state sequence $Q$, and there are many possible state paths that can generate the same observation sequence. Therefore, the joint probability in the above equation is summed over all possible state sequences.

The $N$-state left-right model starts at state 1 and end in state $N$ at the time step $T$. However, the first and last states in our model are the non-emitting states. The state sequence can be extended to cover these non-emitting states. For example, consider the particular state sequence $Q^T = \{q_0, q_1, ..., q_T, q_{T+1}\}$ where $q_t$ is the state at time $t$, $q_0 = s_1$, and $q_{T+1} = s_N$. The $P(\mathbf{O}^T, Q^T|\lambda)$ can be expressed as:

$$\begin{aligned} P(\mathbf{O}^T, Q^T|\lambda) &= P(Q^T|\lambda)\, P(\mathbf{O}^T|Q^T, \lambda) \\ &= \prod_{t=1}^{T+1} a_{q_{t-1}q_t} \prod_{t=1}^{T} b_{q_t}(\mathbf{o}_t). \end{aligned} \tag{3.6}$$

This makes the computation of Eq. 3.5 very expensive since it requires the order of $2TN^T$ multiplications. A more efficient calculation can be done by introducing the *forward probabilities*. The forward probability, $\alpha_i(t)$, is defined as the probability that the partial observation sequence $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t$ is observed and the model is in state $s_i$ at time $t$, i.e.

$$\alpha_i(t) = P(\mathbf{o}_1, ..., \mathbf{o}_t, q_t = s_i|\lambda). \tag{3.7}$$

The computation of the forward probabilities can be performed recursively according to the following steps:

- Initialisation

$$\alpha_j(0) = \begin{cases} 1 & \text{for } j = 1 \\ 0 & \text{for } 1 < j < N \end{cases} \tag{3.8}$$

- Recursion, for $1 \le t \le T$

$$\begin{aligned} \alpha_1(t) &= 0 \\ \alpha_j(t) &= \left[ \sum_{i=1}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\mathbf{o}_t) \quad \text{for } 1 < j < N. \end{aligned} \tag{3.9}$$

The likelihood in Eq. 3.5 is computed from the forward probabilities by:

$$P(\mathbf{O}^T|\lambda) = \alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T) a_{iN}, \tag{3.10}$$

which significantly reduces the number of computations. Alternatively, the above likelihood can be computed from the the *backward probabilities*, $\beta_i(t)$. The backward probability is defined as the probability that the partial observation sequence $\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, ..., \mathbf{o}_T$ is observed, from time $t+1$ to $T$, given that the model is in state $s_i$ at time $t$, i.e.

$$\beta_i(t) = P(\mathbf{o}_{t+1}, ..., \mathbf{o}_T | q_t = s_i, \lambda). \tag{3.11}$$

In a similar manner, the backward probability can be recursively computed starting from the end of the observation sequence as follows:

- Initialisation

$$\beta_i(T) = \begin{cases} 1 & \text{for } i = N \\ a_{iN} & \text{for } 1 < i < N \end{cases} \tag{3.12}$$

- Recursion, for $1 \le t < T$

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_j(t+1) \quad \text{for } 1 < i < N. \tag{3.13}$$

This yields the alternative likelihood computation based on the backward probabilities as follows:

$$P(\mathbf{O}^T|\lambda) = \beta_1(0) = \sum_{j=2}^{N-1} a_{1j} b_j(\mathbf{o}_1) \beta_j(1). \tag{3.14}$$

Both the forward and backward probabilities are the basic variables useful for the computation of other important probabilities required by the HMM training process.

## 3.3 Maximum Likelihood Training in HMMs

By defining the properties of the HMM, one of the most important questions arises: how can we obtain the models that implement the Bayes decision theory? This problem is known as the HMM training problem or the estimation of the HMM parameters based on the observed samples. The most commonly used method for HMM training is Maximum Likelihood (ML) estimation, which aims at adjusting the model parameters so as to maximise the probability of the training samples generated by the model. This is equivalent to maximising the ML objective function in Eq. 2.14. Consider a set of $R$ training samples $\mathbf{O} = \{\mathbf{O}_1^T, ..., \mathbf{O}_r^T, ..., \mathbf{O}_R^T\}$ and their correct categories $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^r, ..., \mathcal{C}^R\}$, where $\mathcal{C}^r \in \{C_1, ..., C_M\}$. Each sample $\mathbf{O}_r^T$ is a sequence of $T$ observation outputs, $\{\mathbf{o}_{r,1}, ..., \mathbf{o}_{r,t}, ...\mathbf{o}_{r,T}\}$. The overall HMM parameters, $\Lambda$, are composed of $M$ models, i.e. $\Lambda = \{\lambda_1, ..., \lambda_M\}$, where $\lambda_m$ denotes the parameters of the HMM representing the category $C_m$. The ML objective function can be rewritten as:

$$f_{ML}(\Lambda) = \sum_{r=1}^{R} \log\, P_\Lambda(\mathbf{O}_r^T|\mathcal{C}^r). \tag{3.15}$$

The maximisation of the above equation is achieved by adjusting the model parameters of each category separately so that its likelihood is maximised. Note that the parameters of a particular model are estimated from the samples belonging to that model. Each model parameter can be optimised by setting the derivative of $f_{ML}$ with respect to that parameter to zero. However, optimising the complex likelihood function of the HMMs is analytically intractable. An efficient iterative procedure known as the Baum-Welch algorithm [45] is regarded as a conventional method to estimate the HMM parameters and one of the most important reasons for the popularity of ML training in the HMMs. The algorithm is mathematically guaranteed to converge, at least to the local maximum, and requires only few iterations to obtain the desired results [30].

The Baum-Welch algorithm is derived from the Expectation-Maximisation (EM) algorithm. The ensuing sections describe the basic idea behind the EM algorithm followed by one of its instances, the Baum-Welch algorithm, for the estimation of the HMM parameters.

### 3.3.1 Expectation-Maximisation Algorithm

The Expectation-Maximisation (EM) algorithm is a general method for ML estimation where the observed data are incomplete or involve some missing values. It is also useful to optimise the likelihood function that is analytically complicated. The theory of the EM algorithm can be found in [14, 48]. This section focuses on its basic concept to gain a better understanding of the Baum-Welch algorithm, to be discussed in the next section. The fundamental idea of the EM algorithm involves the introduction of the hidden variables in the estimation process which help simplifying the optimisation of the likelihood function. The parameter estimation in the EM algorithm is performed iteratively via the use of an auxiliary function $\mathcal{Q}(\lambda, \lambda^\tau)$, where $\lambda^\tau$ is the current model parameters treated as a constant, and $\lambda$ represents the new parameter set that needs to be estimated. This auxiliary function has the following important property: the particular estimate of $\lambda$ (for example, $\lambda^{\tau+1}$) which increases the value of $\mathcal{Q}$ is bound to increase the likelihood of the observed data $\mathbf{x}$, i.e. if $\mathcal{Q}(\lambda^{\tau+1}, \lambda^{\tau}) \geq \mathcal{Q}(\lambda^{\tau}, \lambda^{\tau})$ then $P(\mathbf{x}|\lambda^{\tau+1}) \geq P(\mathbf{x}|\lambda^{\tau})$.

The auxiliary function employed in the EM algorithm is written in the following form:

$$\begin{aligned}\mathcal{Q}(\lambda, \lambda^{\tau}) &= E_Q\left[\log P(\mathbf{x}, Q|\lambda) \,|\, \mathbf{x}, \lambda^{\tau}\right] \\ &= \sum_{Q} \log P(\mathbf{x}, Q|\lambda) \; P(Q \,|\, \mathbf{x}, \lambda^{\tau}),\end{aligned} \tag{3.16}$$

which is the expected value of the log-likelihood, $\log P(\mathbf{x}, Q|\lambda)$, with respect to the introduced hidden variable $Q$ given the observed data $\mathbf{x}$ and current model parameters $\lambda^{\tau}$. The process of the EM algorithm begins with the initialisation of model parameters $\lambda^{\tau}$. Then two iterative steps are carried out: the first step (the E-step) calculates $\mathcal{Q}$ in Eq. 3.16 for the given $\mathbf{x}$ and $\lambda^{\tau}$; and the second step (the M-step) is to maximise $\mathcal{Q}$. The parameter set $\lambda$ that maximises $\mathcal{Q}$ is selected as the initial parameter set for the next iteration. These processes are repeated until the likelihood $P(\mathbf{x}|\lambda)$ converges. The algorithm is guaranteed to converge at least to the local maximum of the likelihood function, and the final $\lambda$ is the estimate of the model parameters.

## 3.3.2 ML Estimation of HMM parameters

Consider the observed sample $\mathbf{O}_r^T$ and let its correct category $\mathcal{C}^r$ be represented by the model $\lambda$. The ML objective function in Eq. 3.15 aims at maximising $\sum_{r=1}^{R} P(\mathbf{O}_r^T|\lambda)$ . The estimation of $\lambda$ is performed category-by-category by using the samples of that category and the Baum-Welch algorithm, which is derived from the EM algorithm for this purpose. Given the fact that $P(\mathbf{O}_r^T|\lambda^{\tau})$ is independent of $\lambda$, the auxiliary function in Eq. 3.16 becomes:

$$\mathcal{Q}(\lambda, \lambda^{\tau}) = \sum_{Q} \log P(\mathbf{O}_r^T, Q|\lambda) \; P(\mathbf{O}_r^T, Q|\lambda^{\tau}), \tag{3.17}$$

where, in this case, the hidden variables $Q$ are the state sequences. By substituting Eq. 3.6, the above equation can be written as:

$$\mathcal{Q}(\lambda, \lambda^{\tau}) = \sum_{Q} P(\mathbf{O}_r^T, Q|\lambda^{\tau}) \sum_{t=1}^{T+1} \log a_{q_{t-1}q_t} + \sum_{Q} P(\mathbf{O}_r^T, Q|\lambda^{\tau}) \sum_{t=1}^{T} \log b_{q_t}(\mathbf{o}_t). \tag{3.18}$$

Note that the optimisation problem becomes less complex because the parameters to be estimated are independently split into different terms.

Consider the first term in Eq. 3.18:

$$\sum_{Q} P(\mathbf{O}_r^T, Q|\lambda^{\tau}) \sum_{t=1}^{T+1} \log a_{q_{t-1}q_t} = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=1}^{T+1} P(\mathbf{O}_r^T, q_{t-1} = s_i, q_t = s_j|\lambda^{\tau}) \log a_{ij}. \tag{3.19}$$

The estimation of the transition probabilities, $a_{ij}$, can be done by setting the derivative of Eq. 3.19 with respect to $a_{ij}$ to zero, and imposing the constraint $\sum_{j=1}^{N} a_{ij} = 1$, $\forall i$. This yields the following formula for the estimation of the transition probabilities, $\hat{a}_{ij}$, when using all training samples [48]:

$$\begin{aligned}\hat{a}_{ij} &= \frac{\sum_{r=1}^{R}\sum_{t=2}^{T} P(\mathbf{O}_r^T, q_{t-1} = s_i, q_t = s_j|\lambda^\tau)}{\sum_{r=1}^{R}\sum_{t=1}^{T} P(\mathbf{O}_r^T, q_t = s_i|\lambda^\tau)} \\ &= \frac{\sum_{r=1}^{R}\sum_{t=2}^{T} \xi_{ij}^r(t-1)}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_i^r(t)}.\end{aligned} \tag{3.20}$$

$\xi_{ij}^r(t)$ is the probability of the model being in state $s_i$ at time $t$ and making a transition to state $s_j$ at time $t+1$, $P(\mathbf{O}_r^T, q_{t-1} = i, q_t = j|\lambda^\tau)$. On the other hand, $\gamma_i^r(t)$ represents the probability of the model being in state $s_i$ at time $t$, $P(\mathbf{O}_r^T, q_t = i|\lambda^\tau)$. These two probabilities can be calculated by using the forward and backward probabilities as follows:

$$\begin{aligned}\xi_{ij}^r(t) &= \frac{\alpha_i^r(t) a_{ij} b_j(\mathbf{o}_{r,t+1}) \beta_j^r(t+1)}{p(\mathbf{O}_r^T|\lambda^\tau)} \text{ , and} \\ \gamma_i^r(t) &= \frac{\alpha_i^r(t)\beta_i^r(t)}{p(\mathbf{O}_r^T|\lambda^\tau)}.\end{aligned} \tag{3.21}$$

The estimation of the output probabilities, $b_j$, can be done by considering the second term of Eq. 3.19. Since the parameters of each mixture component need to be optimised in isolation, an additional hidden variable is introduced to indicate a mixture component that generates an output at each time. The resulting formulae to update the Gaussian parameters are derived as:

$$\begin{aligned}\hat{c}_{jm} &= \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_{jm}^r(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_j^r(t)}, \\ \hat{\boldsymbol{\mu}}_{jm} &= \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_{jm}^r(t)\mathbf{O}_r^T}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_{jm}^r(t)}, \\ \hat{\boldsymbol{\Sigma}}_{jm} &= \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_{jm}^r(t)(\mathbf{O}_r^T - \hat{\boldsymbol{\mu}}_{jm})(\mathbf{O}_r^T - \hat{\boldsymbol{\mu}}_{jm})'}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_{jm}^r(t)},\end{aligned} \tag{3.22}$$

where $\gamma_{jm}^r(t)$ denotes the probability of occupying the $m$-th mixture component in state $s_i$ at time $t$ and is sometimes referred to as an *occupancy count*. It is computed by:

$$\gamma_{jm}^r(t) = \frac{\sum_{i=1}^{N-1} \alpha_i^r(t-1) a_{ij} c_{jm} b_{jm}(\mathbf{o}_{r,t}) \beta_j^r(t)}{p(\mathbf{O}_r^T|\lambda^\tau)}. \tag{3.23}$$

Note that the Baum-Welch algorithm uses the forward and backward probabilities to calculate other related probabilities. The algorithm is also known as the forward-backward algorithm.

## 3.4 Applications of HMMs to Handwriting Recognition

HMMs have been applied to handwriting recognition problems over decades. Although other techniques (such as neural networks and the prototype-based classifiers) can be used, the HMMs are deemed most appropriate to cursive handwriting recognition because of their ability to model continuous signals. A summary of the previous applications of the HMMs to handwriting recognition is provided in the following sections. It is categorised into the field of on-line and off-line handwriting recognition. These applications employed ML training for the estimation of their model parameters.

### 3.4.1 HMM-Based On-line Handwriting Recognition

On-line handwriting can be viewed as a generation of signals (x-y coordinates) over time. It is similar to the speech signal to some extent, although the features used to represent both signals are relatively different. According to this similar property, the HMMs are suitable to the problem of on-line handwriting recognition, and their previous applications can be summarised as follows.

Nag *et al.* [49] employed HMMs for the recognition of handwritten words, 'one' to 'ten'. Each word was modelled by an HMM with discrete output densities. This work was among the first to demonstrate the potential of the HMMs applied to on-line handwriting recognition. A similar approach was utilised in [50]. While the method of representing one word by one HMM is practical for small vocabulary systems, it is inefficiently applicable to large vocabulary systems due to the growing number of models in respect to the number of vocabularies. Later work attempted to solve this problem by modelling the handwritten words at the character level. For example, in the research of Nathan *et al.* [51], continuous density HMMs were used to recognise characters written in isolation. The word models are constructed by concatenating the corresponding character models. In their following work [52], the character models were efficiently applied to the large vocabulary recognition task (21,000-word lexicon). The use of character models has become increasingly popular as evident from its applications in the context dependent recognition systems [53] and very large vocabulary recognition systems [54], with the help of statistical grammar and language models. Alternatively, Hu *et al.* [55] introduced an approach by using sub-character or stroke models which can be shared among several character models. Such an approach has demonstrated good performance in many applications [56], including the recognition of large vocabulary writer independent on-line handwriting [57]. Sub-stroke HMMs have become an important candidate for Chinese [58], Japanese [59] and Korean [60] character recognition where their characters are normally composed of multiple strokes.

### 3.4.2 HMM-Based Off-line Handwriting Recognition

The ability to model stochastic processes makes HMMs the attractive candidates for off-line handwriting recognition which needs to cope with shape variations and cursiveness which occurred in handwriting. Nevertheless, off-line handwriting is represented as a static image that does not convey time-varying signals suitable for HMM modelling. To overcome this problem, the sliding-window technique (described in Section 6.3) is commonly used to generate an observation sequence from the handwriting image. Another technique [61] produces a sequence of arcs from the skeleton of word image. This technique requires a skeletonisation process which is often susceptible to noises and geometrical transformations. Both discrete and continuous density HMMs have been found in the literature. The characteristics of features are determined by the type of the HMMs. The discrete HMMs [62] use the stroke shapes, descenders and ascenders in a sliding window as the feature vectors whereas the continuous approach [63] usually extracts the features from the pixel values. Thus far, several applications have benefitted from the HMM-based recognition systems, such as isolated character recognition, bank check reading, postal address recognition and general handwritten text recognition. A comprehensive survey of these applications can be found in [64, 65].

Using the HMMs to model handwriting at the word level is useful when there are a small number of words to be recognised, as evident in [66]. However, modelling the words at the character level is proved more practical since the character models can be shared among several words, thereby significantly reducing the number of model parameters. Over the past few years, several attempts have been made to apply HMMs to large vocabulary recognition, as seen in the recent survey by Koerich *et al.* [67]. Recent work by Koerich *et al.* [68] and Vinciarelli *et al.* [4] demonstrated some promising results of the HMM-based large vocabulary recognition systems obtained through the support of statistical language models.

## 3.5 Discriminative Training in HMMs

ML training in the HMMs aims at individually modelling each category in order to maximise the probability of the training samples being generated by the model (see Section 3.3). Only the samples belonging to the category are used to train the corresponding model. On the contrary, the discriminative training approach (as discussed in Chapter 2) focuses on separating the models from one another by taking into account all samples of competitive categories. Discriminative training has demonstrated an improvement of recognition accuracies from ML training in many HMM-based speech recognition applications. However, the applications of discriminative training in HMMs in other areas, including character recognition, are scant. Biem revealed significant evidence of discriminative training in the HMMs for handwriting recognition where MCE training was applied to the recognition of isolated handwritten characters [42] and on-line handwritten words [43]. Reductions in error rates of more than 30% in [42] and 17% in [43] were achieved when compared to ML training. The discriminative training approach in Part II of this dissertation centres on the MMI criterion applied to HMM-based handwriting recognition systems, which has been under-researched in previous work.

MMI training aims to find a parameter set of the HMMs which maximises mutual information between the samples and their correct categories. This can be done by optimising the MMI objective function. Given similar notations as in Section 3.3, the MMI objective function from Eq. 2.21 can be rewritten here as:

$$f_{MMI}(\Lambda) = \frac{1}{R}\sum_{r=1}^{R}\left(\log P_\Lambda(\mathbf{O}_r^T|\mathcal{C}^r) - \log\sum_{i=1}^{M} P_\Lambda(\mathbf{O}_r^T|C_i)P(C_i)\right). \tag{3.24}$$

Optimising $f_{MMI}$ involves the maximisation of the difference of two terms: the conditional probabilities of the training samples given their correct categories and the corresponding conditional probabilities of all competitive classes. The second term makes MMI training discriminative and differs from ML training. One reason behind the popularity of ML training in the HMMs is the powerful Baum-Welch algorithm. Nonetheless, such an algorithm is not applicable to the optimisation of $f_{MMI}$, and, perhaps, this has prevented the extensive usage of MMI training in character recognition. Despite this fact, the optimisation algorithms for MMI training have been constantly improved over the past decades and recently reached their maturity stage. The following sections discuss some of the problems encountered during MMI training and summarise two important algorithms for the optimisation of $f_{MMI}$.

### 3.5.1 Gradient-Based Optimisation Methods

Although the standard Baum-Welch algorithm is an efficient algorithm for ML training in HMM, it cannot be applied to the optimisation of the MMI objective function (see in Section 3.5.2). Therefore, in the early stage, the research on MMI training mainly relied on the gradient-based methods for this optimisation [18, 21, 69]. A number of gradient-based techniques are compared in [30, 70]. The gradient-based approach is a general iterative method for finding an estimate that either minimises or maximises the function. This involves the calculation of gradients which is subsequently used to determine the degree of parameter adjustments. The gradient descent (GD) algorithm is the simplest method of the gradient-based approach. Note that the term *descent* is always used, even though, in practice, the algorithm can be applied to the maximisation problem where the term *ascent* seems to be more appropriate. The GD algorithm applied to MMI training operates in the following manner. The optimisation begins with the initialised model parameters $\Lambda$. In each iteration, the parameter adaptation is performed according to:

$$\hat{\Lambda} = \Lambda + \eta\,\nabla f_{MMI}, \tag{3.25}$$

which processes in a hill-climbing manner. The parameters are modified in the direction specified by the gradient of the MMI objective function, whereas the magnitude of an update is a constant $\eta$, known as the learning rate, times the magnitude of this gradient.

Although the GD algorithm is a general technique that is guaranteed to locate the local maximum, its main disadvantage is the slow speed of convergence, which may take up to infinite iterations, especially when applied to the complex MMI objective function of the HMMs. Several techniques have been proposed to improve its convergence rates, for example, the introduction of the momentum factor and the Newton's method that incorporates the use of the second-order derivatives (i.e. the Hessian matrix, $\nabla^2 f_{MMI}$). These alternative methods are, however, impractical for the MMI estimation because the appropriate momentum tuning is difficult to achieve and the Hessian matrix requires substantial computations. In this research, we rely on the approximate second order method called QuickProp [29] which simplifies the calculation of the Hessian matrix based on a number of assumptions. Section 4.4 explains the QuickProp algorithm in more detail and also provides the evaluation of both the GD and QuickProp algorithms in MMI training.

### 3.5.2 Extended Baum-Welch Algorithm

The MMI objective function in Eq. 3.24 is expressed as a difference of two terms: the likelihood for the correct category and the sum of the likelihoods of all categories. In [10, 30], these likelihoods are referred to as the *numerator* and *denominator* likelihoods, respectively. The negative sign in the denominator likelihood prohibits the use of the Baum-Welch algorithm in MMI training. As opposed to Section 3.3.1, an increase in the auxiliary function $\mathcal{Q}$ does not imply an increase in the MMI objective function.

Much effort has been made to extend the Baum-Welch algorithm to MMI training. Prior to discussing each technique, let us consider the following general formula for updating the transition probabilities $a_{ij}$ by ML (obtained by solving the auxiliary function in Eq. 3.16):

$$\hat{a}_{ij} = \frac{a_{ij}\,\frac{\partial f_{ML}}{\partial\, a_{ij}}}{\sum_{k=1}^{N} a_{ik}\,\frac{\partial f_{ML}}{\partial\, a_{ik}}}. \tag{3.26}$$

Eq. 3.26 can be applied to all parameters of discrete output HMMs and some CDHMM's parameters which are subject to the sum-to-one property ($\sum_{j=1}^{N} a_{ij} = 1$), such as the mixture coefficients, $c_{jm}$. Gopalakrishnan *et al.* [31] proposed an extension of the Baum-Welch algorithm, called the Extended Baum-Welch (EBW) algorithm, which is applicable to the discrete output HMMs. In that work, a positive constant $D$ is introduced to Eq. 3.26 and the re-estimation formula for MMI training becomes:

$$\hat{a}_{ij} = \frac{a_{ij}\left(\frac{\partial f_{MMI}}{\partial\, a_{ij}} + D\right)}{\sum_{k=1}^{N} a_{ik}\left(\frac{\partial f_{MMI}}{\partial\, a_{ik}} + D\right)}. \tag{3.27}$$

The smoothing constant $D$ is set to ensure that all updated values $\hat{a}_{ij}$ are positive and, thus, is dominated by the small-valued parameters. Even though the value of $D$ is chosen as small as possible, the resulting value of $D$ can become so large that the convergence rate is too slow to be useful.

Normandin *et al.* [71] modified the EBW algorithm, based on the idea of [69], to improve convergence. The approximate method, which focuses more on high-valued parameters, yet still relies on the constant $D$, was used in their studies (see Eq. 4.24). Although their approach lacks theoretical foundations, its approximation demonstrates good convergence in practice. Normandin [32] also successfully applied the EBW algorithm to the continuous density HMMs. This derivation can be found in [72]. Over the recent years, the EBW algorithm has gained substantial recognition in MMI training. Valtchev [33] and Schluter [73] found this extension very useful.

The original proof of the EBW's validity in [71] is satisfied only with the infinite value of $D$, whereas the finite value of $D$ is used in practice. Further research has been carried out in order to find a better justification for this validity. Schluter *et al.* [74] and Zheng *et al.* [75] showed that the re-estimation equations equivalent to EBW can be derived from the gradient-based approaches when the appropriate smoothing constants and learning rates are provided. However, these approaches are difficult to apply to the estimation of parameters with the sum-to-one property. Recently, Povey [10] has proposed a more efficient approach to derive the EBW formulae based on the notion of *weak-sense* auxiliary functions, the functions that have the same gradient as the objective function around the particular parameter values (see Section 6.4.1). He successfully proves the convergence of EBW for the finite-$D$ case and suggests an efficient method to adjust $D$. Moreover this approach provides a comprehensive way to derive the update formulae for means and variances of Gaussians. The novel approach to the optimisation of mixture coefficients and transition probabilities was proposed and proved efficient. Complete derivations for this approach can be found in [10]. Summaries for these formulae are given in Section 6.4.2.

## 3.6 Applications of MMI Training to HMMs

Recent applications of discriminative training are briefly reviewed in Section 2.5. This section presents a detailed summary of the previous research of MMI training applied to the HMMs in particular. The applications discussed here are entirely in the field of speech recognition, but insights from speech recognition can be useful in handwriting recognition.

Bahl *et al.* [18] are among the first authors who publish the results of MMI training. They applied MMI training to the speaker dependent isolated word recognition system. The output probabilities of the HMMs were modelled by the discrete probability densities. In their work, a relative improvement in word error rates of 18% was reported, in comparison to ML training. Brown [21] revealed comparable results in the phoneme recognition application using continuous density HMMs. Merialdo [69] successfully applied MMI training to the continuous phoneme recognition for French using discrete HMMs. An improvement of approximately 23% relative was obtained in his work. These previous studies on MMI training [18, 21, 69] relied on the GD approach to optimise the MMI objective function and suffered from the slow speed of convergence. In [69], Merialdo found that the gradients were dominated by low valued parameters. Therefore, he proposed an approximation method that emphasised more on high-valued parameters. Although his technique was not guaranteed to increase the objective function in every iteration, the rate of convergence was significantly improved in general.

Gopalakrishnan *et al.* [31] proposed an optimisation algorithm, which was derived from the Baum-Welch algorithm, applicable to the rational function optimisation. The MMI objective function falls into this category because it can be written as a ratio of two polynomials: the numerator and denominator likelihoods. Such an algorithm is known as the Extended Baum-Welch (EBW) algorithm, which was applied to the discrete HMMs and demonstrated better convergence than the GD algorithm. Since then, the EBW algorithm has attracted many researchers in MMI training, even though the tuning of smoothing parameters caused some difficulties. The work done by Normandin *et al.* [32] provided significant evidence for the efficiency of MMI training in the connected digit recognition tasks. The EBW algorithm was modified by following Merialdo [69]'s approach to improve the convergence rate. Although the convergence was not guaranteed at every iteration, the method was practically useful.

There were a number of studies that still relied on the gradient-based approaches with the help of second-order derivatives and their approximations to improve the convergence rate. Kapadia [70] and Valtchev [30] made comprehensive comparisons for these methods, including EBW, where MMI training was performed on various standard speech phoneme databases. The QuickProp algorithm, provided faster convergence than other approaches. Nonetheless, Valtchev [30] suggested that their performance suspiciously depended on the recognition tasks. Regarding the development of EBW, Normandin [32] formulated update equations for the continuous density HMMs by mapping them to the discrete distributions in the limited condition. Their techniques have been widely used and yielded good results (i.e. a relative reduction in recognition errors of 40% compared to ML training). In the later work of Valtchev *et al.* [33], similar EBW update formulae were utilised. It is regarded as the first to demonstrate the utility of MMI training in the continuous density HMMs for large vocabulary recognition. Recently, Povey [10] has extended previous work by changing and improving the training algorithms. Many aspects of the training procedures were experimentally investigated in his work and Schluter's work [73].

Despite the success of EBW in MMI training, the proof for its convergence for the continuous case remains controversial. The convergence-guaranteed condition is obtained when an infinite value of smoothing parameters is used [71], whereas the finite value is utilised in practice. A search for better justification of the EBW algorithm has been continuously pursued. Schluter *et al.* [74] and Zheng *et al.* [75] showed that, given the appropriate smoothing constants and learning rates, the update formulae equivalent to the ones used in EBW can be derived from the traditional GD approach. This proof does not rely on the aforementioned condition. However, their techniques can only apply to the Gaussian parameters, not to the mixture coefficients and transition probabilities that are subject to a sum-to-one constraint. Alternatively, Povey [10] introduced the concept of *strong-sense* and *weak-sense* auxiliary functions which provided an efficient way to derive the EBW update rule valid for any values of smoothing parameters. His approach also led to the novel update rules for the sum-to-one parameters (see Section 6.4.2).

Computational requirements for MMI training is one of the important issues to be considered because the likelihoods of all competing models need to be calculated. The problem becomes more severe in the continuous speech recognition tasks where several combinations of word sequences are available. Chow [76] used $N$-best lists to speed up computation by considering only the $N$ most likely candidates when computing the denominator likelihoods. However, this technique becomes inefficient for very long sentences. A much more efficient way is to use a *lattice* which represents word sequences in the form of graph. Valtchev *et al.* [33] and Povey [10] found this approach very computationally effective for MMI training of large vocabulary systems while the recognition performance remained unaffected. Note that, however, the lattices are equivalent to the $N$-best lists in the case of the isolated phoneme or word recognition.

## 3.7 Summary

This chapter provides background knowledge of the Hidden Markov Model (HMM), including theoretical perspectives and training algorithms used to estimate its parameters. The HMMs have been extensively deployed in handwriting recognition applications, both on-line and off-line. In these applications, their parameters are estimated by Maximum Likelihood (ML) training through a powerful optimisation algorithm, namely the Baum-Welch algorithm. A literature survey shows that discriminative training has improved the recognition performance of many speech recognition systems when compared to ML training. However, there has been scant empirical evidence of its success in attempting the handwriting recognition problem. The second half of this chapter is devoted to the concept of discriminative training with the emphasis on Maximum Mutual Information (MMI) training in the HMMs and its optimisatisation algorithms. MMI training is intended to adjust the model parameters to maximise the mutual information between the samples and their correct categories in order to separate all categories effectively. Despite these facts, it is regarded as an ongoing research area, and many of its aspects have not been explored. One important issue is the study of its performance in a highly constrained system where the parameters are shared across several models, and the optimisation of the parameters in one model has an impact on those of other models. Chapter 4 presents an in-depth study of this topic and introduces MMI training to the HMM-based handwriting recognition system.

# Chapter 4

# MMI Training of Tied-Mixture Density Hidden Markov Models

## 4.1 Introduction

Discriminative training in the HMM-based recognisers, as discussed in Chapter 3, has been effectively applied to speech recognition problems and improved recognition rates over conventional ML training. Nevertheless, discriminative training is an ongoing research area, and many of its aspects have not been explored. Much effort is also required to evaluate its performance in character recognition tasks. Moreover, the majority of discriminative training successes have been acknowledged in the context of continuous density HMMs (CDHMMs). Such a recognition structure, where models are well-separated from each other, is relatively appropriate for the discriminative training scheme. In the HMM-based framework, the method of parameter tying can significantly reduce the number of parameters in the system so as to cope with a limited amount of training data. It remains unclear whether discriminative training in a highly-tied system, such as the system based on tied-mixture density HMMs (TDHMMs), is as effective as that in the CDHMM-based system.

Because the applications of MMI training in the HMMs to handwriting recognition problems are scant in the literature, this chapter aims to introduce MMI training to the TDHMM-based handwriting recognition system and evaluates its performance on the unconstrained-style on-line handwritten digit recognition task. This multi-writer digit recognition task provides a suitable framework to explore the effectiveness of MMI training in this context because it consists of a reasonable number of classes while maintaining a wide variety of handwriting styles. The recognition performance is compared with both the baseline ML training results and the results of MMI training in the CDHMM system. A contribution of the present study includes finding the most appropriate optimisation algorithm for MMI training in this particular task, and determining the best set of optimised parameters that yields maximum discriminative abilities.

Section 4.2 describes an on-line handwritten digit database and an overview of the recognition system used in this research. Our recognition system operates in two sequential stages: the *fast-match* and *detailed-match* stages. Section 4.3 introduces a concept of parameter tying in HMMs. The re-estimation formulae of various optimisation algorithms for MMI training in the TDHMM context are derived and evaluated in Section 4.4. The experiments in Section 4.5 are conducted to determine which sets of model parameters contribute to the best performance of MMI training in the TDHMM system. In order to test the hypothesis that parameter tying may inhibit the discriminative ability of the MMI algorithm, the experiments of MMI training in the CDHMM system which does not have a parameter-sharing property are investigated in Section 4.6. The experiments of MMI training in Sections 4.4, 4.5, and 4.6 are performed based on the fast-match stage. Section 4.7 evaluates the performance of MMI training in the detailed-match stage. The discussion and summary of the chapter are given in Section 4.8.

## 4.2 Recognition System Overview

The database and the feature extraction technique used in this research are described in Section 4.2.1, followed by an overview of the recogniser structure in Section 4.2.2.

### 4.2.1 Database and Feature Extraction

In this chapter, the digit recognition application is chosen as our recognition problem. Although this is a relatively simple recognition task, the unconstrained-style handwriting makes it more difficult. Supplied with 10 well-separated output categories, it provides a suitable framework for our studies of MMI training and is regarded as an important test case for more general recognition problems. The on-line handwritten digit database in this study is a subset of the database used in the work of Nathan *et al.* [52] in which approximately 100 writers were asked to write on the digitised tablets with no constraints imposed. A subset of 500 samples per digit was randomly selected for training and 350 samples for testing. As a result, the database containing 5,000 training samples and 3,500 testing samples was constructed.

On-line handwriting was sampled at 100Hz frequency and stored as a sequence of $(x, y)$-coordinate sample points over time. The pre-processing stage discussed in [77] was required to normalise the size of characters and to interpolate those points so that they were equally spaced. This removed the inconsistency of different writing speeds. The feature vector, $\mathbf{F}$, was extracted from each point $(x, y)$ along the trajectory, where $\mathbf{F}$ is a 5-dimensional feature vector composing of $\Delta x$, $\Delta y$, $\sin\theta$, $\cos\theta$ and the relative value of $y$ with respect to the baseline. The $\theta$ (an angle) and $\Delta$-operator are measured between two adjacent points. Consequently, the character was represented by a sequence of 5-dimensional feature vectors. However, this representation was still not appropriate for the recogniser since it contained a large number of feature vectors for each character. To reduce the number of feature vectors, a pre-defined number $(P)$ of feature vectors were combined to form one large feature vector ($5P$ dimensions), or *frame*. One such character usually contains 3 to 9 frames. Principal Component Analysis (PCA) (see Section 6.3.2) was then applied to project a high-dimensional vector into a compact feature vector $\mathbf{o}_t$, which represented each $t$-th frame. Detailed discussions of this feature extraction technique can be found in [52, 78].

### 4.2.2 Recogniser Structure

The HMM-based recogniser used in this chapter is the on-line unconstrained handwriting recognition system developed by IBM researchers. It has been successfully applied to many applications of on-line cursive handwriting recognition [52, 79, 80, 81, 82]. Originally, the recogniser is designed for large-vocabulary real-time handwriting recognition applications, and thus the two-stage recognition structure is employed to reduce computational requirements and recognition time. At the basic level, the recogniser is capable of recognising isolated characters. The recognition process is carried out in two stages: *fast-match* (FM) and *detailed-match* (DM). In the FM stage, simpler models (FM models or single-state HMMs) are used to limit computation and generate a shortlist of likely character candidates. These candidates are passed to the DM stage, where computationally more expensive models (DM models or multi-state HMMs) are used. The DM stage searches the DM models among the shortlisted candidates, and then the most likely model is chosen as the recognised output. A detailed explanation of the system can be found in [79].

The training process is rather complicated because both the FM and DM models are dependent on each other. The FM and DM models are trained by conventional ML training using the Baum-Welch algorithm, as discussed in Section 4.3. The original recogniser does not provide a discriminative training feature. We have developed the MMI training capability into the system which is elaborated in Section 4.4. The following sections briefly discuss the FM and DM stages, based on their training and recognition processes illustrated in Figure 4.1.

#### Fast-Match (FM)

The FM recogniser can be viewed as a collection of single-state HMMs, each of which represents a character model. The FM model uses a mixture of Gaussians as its output probabilities. Since it is composed of single-state HMMs, the FM stage operates on a single frame basis and does not provide duration modelling. It assumes that each frame in the sequence is independent of the others, regardless of its position within the character.

Given the sequence of frames $\{\mathbf{o}_1, ..., \mathbf{o}_t, ..., \mathbf{o}_T\}$ representing the character sample $\mathbf{O}$, the probability that the character model $\lambda_j$ gives rise to the frame $\mathbf{o}_t$ is computed by:

$$\begin{aligned} P(\mathbf{o}_t|\lambda_j) &= \sum_{m=1}^{M} P(\mathbf{o}_t|g_{jm})P(g_{jm}|\lambda_j) \\ &= b_j(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm}\,\mathcal{N}(\mathbf{o}_t, \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}), \end{aligned} \tag{4.1}$$

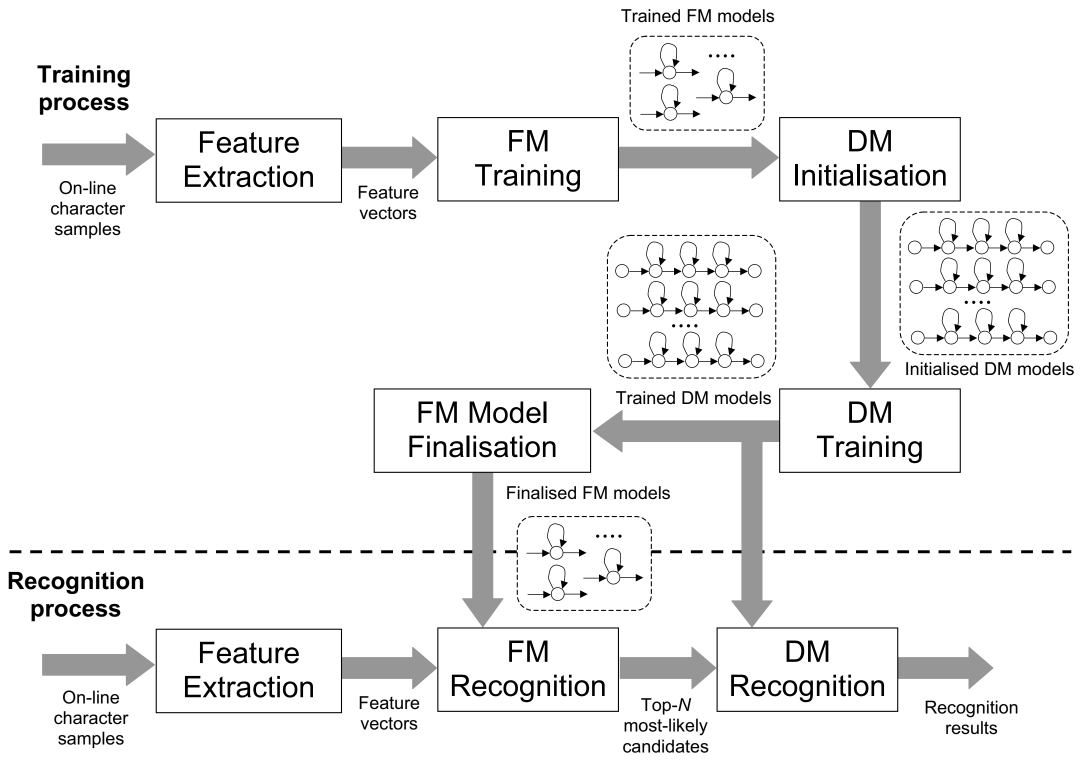


**Figure 4.1:** The training and recognition processes of the two-stage HMM-based recogniser

where $g_{jm}$ is the $m$-th Gaussian mixture of the model $\lambda_j$ and $M$ is the number of mixtures. The output distribution $b_j$ is used to characterise the character class $C_j$. The FM training process estimates the parameters of all Gaussian mixtures, including the means, $\boldsymbol{\mu}_{jm}$, and covariance matrices, $\boldsymbol{\Sigma}_{jm}$, of the Gaussians, and the mixture coefficients, $c_{jm}$, from all frames available in the training samples. The trained FM models are used to initialise the DM models (see the Detailed-Match section).

Given that each frame is independent of the others, in the FM recognition process, the sample $\mathbf{O}$ is recognised as the character $C_i$ when:

$$i = \operatorname*{argmax}_j \prod_{t=1}^{T} P(\mathbf{o}_t|\lambda_j), \tag{4.2}$$

which advocates the category that yields the highest value of the likelihood products. The value of $j$ in Eq. 4.2 varies between 1 and $P$, the total number of character categories. However, the FM recognition stage is not intended to make a final decision of the recognition outputs. Rather, it generates a shortlist of candidates for the DM recognition stage which requires more substantial computations.

### Detailed-Match (DM)

Note that FM modelling excludes a relative position of frames within a character. This notion is introduced in the DM process where each character is modelled by a multiple-state

HMM. Each state associates with an output distribution corresponding to the portion of the character that it models. The DM model is initialised from the corresponding FM model by replicating the single-state model $N$ times, where $N$ is the number of states in the DM models. The Gaussian mixtures from the FM model serve as the initial output distributions for these states. In addition to the self- and next-state transitions, *null* transitions are introduced to the DM model. The null transition does not generate outputs when moving to the next state, as discussed in Section 3.2.1. This only slightly alters the likelihood computation in such a way that some states in the state sequence do not involve output probabilities. A detailed explanation for this computation can be found in [46, 52].

The DM training process re-estimates the model parameters which include the Gaussian parameters, the mixture coefficients and state transition probabilities, by using either conventional ML training or MMI training (see Section 4.4). Because the recogniser is designed for real-time recognition, some computationally expensive calculations are approximated in order to reduce computation [52]. For example, the calculation of likelihood probabilities in the DM models can be approximated by using only the best state sequence obtained from the *Viterbi* path instead of considering all possible state sequences which incorporates forward and backward probabilities, as discussed in Section 3.2.3. In this case, the likelihood in Eq. 3.5 becomes:

$$P(\mathbf{O}^T|\lambda) \approx \max_{Q} P(\mathbf{O}^T, Q|\lambda), \tag{4.3}$$

which requires neither forward nor backward probabilities, thereby making the computation faster. Previous work [52] indicates no significant difference in the recognition rates between the approximated and the actual methods. This approximation is known as Viterbi training and is applied to both ML and MMI training in this chapter.

As previously discussed, both the FM and DM models are dependent on each other. More specifically, the Gaussian mixtures in the FM model are also used by the DM model that represents the same character, and *vice versa*. The DM training process inevitably alters these Gaussian mixtures. Therefore, the finalised FM model is obtained by dividing the trained DM model into a single-state HMM. This requires the mixture coefficients of the FM model to be re-estimated. A recognition procedure is performed sequentially by starting from the FM recognition process, which generates a shortlist of likely character candidates. These candidates are then passed to the DM stage. The DM recognition process is carried out by searching the best-fit DM model among the shortlisted models, rather than making full comparisons across all possible DM models. The most likely model is chosen as a recognition output.

## 4.3 Parameter Tying in HMMs

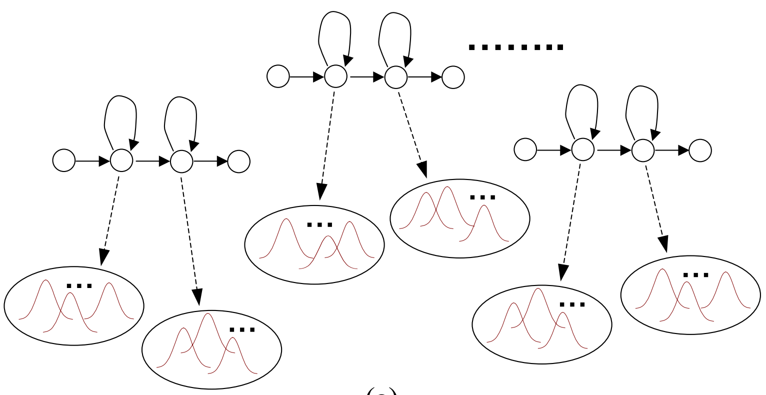
One of the disadvantages of the HMM-based recognisers centres on a large number of model parameters required to be estimated. The number of parameters in the HMMs increases in proportion to these factors: the dimension of feature vectors, the number of states and Gaussian mixtures used in the system. A problem arises when the amount of training data is limited and insufficient to achieve good estimates of these parameters. This results in the models that cannot be well-generalised to the unseen data. A well-known technique to cope with this problem is the employment of parameter tying [82, 83], where parameters are fully or partially shared by all models. As more parameters are tied, the number of model parameters to be estimated decreases. Two important examples of parameter tying in the HMMs are: the tying at state and output distribution levels. In the state-level tying, the idea is to tie together the states which have similar output distributions. This technique is commonly used in speech recognition applications for acoustic modelling, in particular to model some phonemes which are acoustically indistinguishable [84]. On the contrary, in the distribution-level tying, the Gaussian components are tied together to form a Gaussian pool which is then shared among several states and/or models.

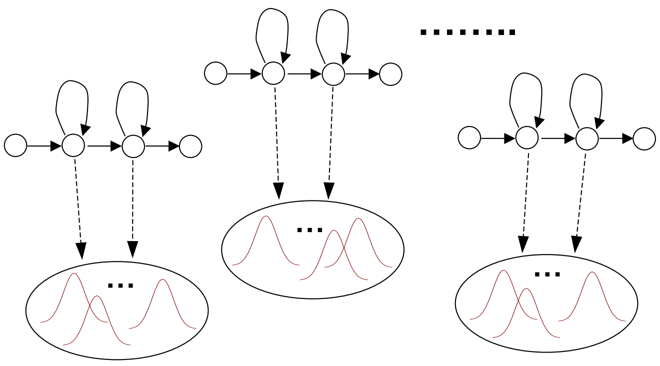
The output-distribution tying is applied to our HMM recogniser. The HMMs employing this tying technique are referred to as the tied-mixture density HMMs (TDHMMs). Because TDHMMs lack modelling capability compared to the fully continuous (CDHMM) system, they are referred to as semi-continuous HMMs. Different approaches can be used to tie together the Gaussian components. The first approach is to start with different Gaussians for each state and merge Gaussians that have similar means and variances. Alternatively, only specific parameters, such as means and variances, are tied across all Gaussians. Another approach is to store all Gaussians as a Gaussian pool which is shared across all state output distributions. The latter approach is applied to our recogniser.

The Gaussian pool in the TDHMMs is called a *codebook*. The codebook can be shared either across all models or within the same model. In the *partially-tied* TDHMM system, each HMM has its own codebook which is shared across the states within that model. In contrast, our recogniser is originally designed to use only one codebook that is shared across all states of all models [52]. This is referred to as the *fully-tied* TDHMM system. It is regarded as the cause of dependencies between the FM and DM models, as discussed in Section 4.2.2.

The difference among the CDHMMs and two types of TDHMMs is illustrated in Figure 4.2. The TDHMMs have an advantage in that the tying process does not affect the convergence properties of the Baum-Welch algorithm [83]. It does not significantly alter the probability computation and the re-estimation formulae. For example, in the case of the fully-tied TDHMMs, $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$ in Eq. 4.1 of the FM stage become $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ because all models share the same Gaussians. Thus, only the mixture coefficients, $c_{jm}$, are model-specific. They are the parameters that characterise the model and discriminate one model from the others.

The majority of the studies with regard to the applications of discriminative training in the HMMs are performed in the context of the CDHMMs and tied-state HMMs [30, 32, 34, 70]. Such recognition structures, where models are well-separated from each other, are deemed appropriate for discriminative training. However, it remains questionable whether discriminative training is effective in a highly-tied system, such as the system based on the fully-tied TDHMMs where the optimisation of Gaussian parameters in one model can effect those of other models. Sections 4.4, 4.5, and 4.6 feature the studies for this evaluation.


(a)


(b)

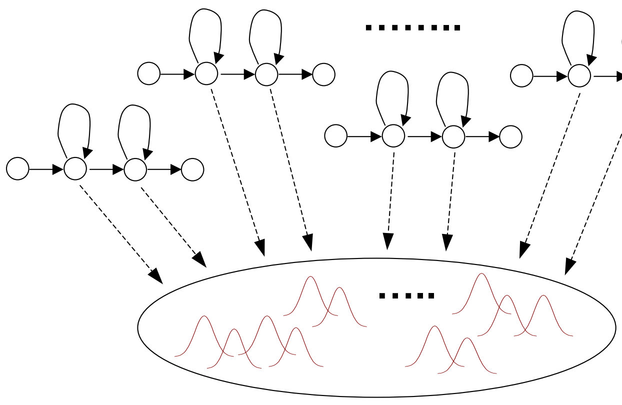

(c)

**Figure 4.2:** Diagrams showing the difference among (a) CDHMMs (b) partially-tied TDHMMs and (c) fully-tied TDHMMs. The diagrams depict multi-state HMMs and their output distributions (pools of Gaussians) which are differently shared across states and models.

## 4.4 MMI Optimisation Algorithms for TDHMMs

MMI training of the HMMs involves adjusting some model parameters to maximise the MMI objective function. Nonetheless, an optimisation algorithm that is as efficient and theoretically proven as the Baum-Welch algorithm for ML training appears to be unavailable for MMI training. This section aims at evaluating a wide range of optimisation algorithms discussed in Section 3.5. It is shown that the performance of optimisation algorithms is dependent on recognition tasks [30]. The optimisation algorithm that yields high convergence rates in one particular task may not necessarily perform well in other tasks. Three optimisation algorithms (i.e., EBW, GD and QuickProp) in the context of TDHMMs are summarised in Sections 4.4.1, 4.4.2 and 4.4.3, respectively. Evaluations of these algorithms are reported in Section 4.4.4.

### 4.4.1 Extended Baum-Welch Algorithm

Consider a set of $R$ training samples $\mathbf{O} = \{\mathbf{O}_1^{T_1}, ..., \mathbf{O}_r^{T_r}, ..., \mathbf{O}_R^{T_R}\}$ and their correct category $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^r, ..., \mathcal{C}^R\}$, where $\mathcal{C}^r \in \{C_1, ..., C_M\}$ and $M$ is the total number of character classes. Each sample $\mathbf{O}_r^{T_r}$ is a sequence of $T_r$ feature vectors (or observation outputs), $\{\mathbf{o}_{r,1}, ..., \mathbf{o}_{r,t}, ...\mathbf{o}_{r,T_r}\}$. Each feature vector $\mathbf{o}_{r,t}$ is composed of $D$ elements (or dimensions), $\mathbf{o}_{r,t} = (o_{r,t,1}, ..., o_{r,t,d}, ..., o_{r,t,D})'$. The model parameters of the HMM representing character $C_i$ are denoted by $\lambda_i$.

#### Mixture Coefficients and State Transition Probabilities

A closed-form solution of the EBW re-estimation formulae in Eq. 3.27 can be applied directly to the parameters that have a sum-to-one property, such as state transition probabilities and mixture coefficients. Given the $k$-th mixture coefficient, $c_{i,j,k}$, of model $\lambda_i$ and state $s_j$, its update ($\hat{c}_{i,j,k}$) can be computed by the EBW algorithm:

$$\hat{c}_{i,j,k} = \frac{c_{i,j,k}\left(\frac{\partial f_{MMI}}{\partial c_{i,j,k}} + D\right)}{\sum_{m=1}^{K} c_{i,j,m}\left(\frac{\partial f_{MMI}}{\partial c_{i,j,m}} + D\right)}, \tag{4.4}$$

where $K$ is the total number of Gaussians in the Gaussian pool. From the above equation, we need to calculate the derivative of $f_{MMI}$ with respect to each $k$-th mixture coefficient, $\frac{\partial f_{MMI}}{\partial c_{i,j,k}}$ (see Appendix A), and specify the value of smoothing parameter $D$. From Eq. A.14, the derivative of the MMI objective function with respect to each mixture coefficient is computed by:

$$\frac{\partial f_{MMI}}{\partial c_{i,j,k}} = \frac{1}{c_{i,j,k}}(\gamma_{i,j,k} - \overline{\gamma}_{i,j,k}), \tag{4.5}$$

where $\gamma_{i,j,k}$ and $\overline{\gamma}_{i,j,k}$ represent the component occupation counts and the *anti-occupation* counts summed over all the data, respectively. These counts are computed by:

$$\gamma_{i,j,k} = \sum_{r=1}^{R}\sum_{t=1}^{T_r} \gamma^r_{i,j,k}(t)\delta(r,i), \tag{4.6}$$

$$\overline{\gamma}_{i,j,k} = \sum_{r=1}^{R}\sum_{t=1}^{T_r} \overline{\gamma}^r_{i,j,k}(t), \tag{4.7}$$

where

$$\delta(r,i) = \begin{cases} 1 & ;\ \mathcal{C}^r = C_i \\ 0 & ;\ \text{otherwise} \end{cases}. \tag{4.8}$$

The term $\gamma^r_{i,j,k}(t)$ in Eq. 4.6 denotes the probability of occupying mixture component $k$ of state $s_j$ in model $\lambda_i$ at time $t$, given the sample $\mathbf{O}_r^{T_r}$. In the Viterbi training approach (see Section 4.2.2), only the best state sequence is taken into account when computing related statistics, rather than considering all possible state sequences. Let $Q_i^r = \{q_1, ..., q_{T_r}\}$ be the sequence of states within the model $\lambda_i$ that yields the maximum probability, given the sample $\mathbf{O}_r^{T_r}$. Hence, the computation of $\gamma^r_{i,j,k}(t)$ requires neither forward nor backward probabilities, as opposed to Eq. 3.23. From Eq. A.8, it is calculated by:

$$\gamma^r_{i,j,k}(t) = \frac{c_{i,j,k}\mathcal{N}(\mathbf{o}_{r,t}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{m=1}^{K} c_{i,j,m}\mathcal{N}(\mathbf{o}_{r,t}, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)} \text{ when } q_t = s_j, \tag{4.9}$$

and $\gamma^r_{i,j,k}(t) = 0$ otherwise. Note that, since the Gaussians in the TDHMMs are tied, they are independent of models and states. Thus, $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ represent the means and covariance matrix of the $m$-th Gaussian among $K$ Gaussians in the pool, respectively. By referring to Eq. A.12, the anti-occupation count for each sample in Eq. 4.7 is computed by:

$$\overline{\gamma}^r_{i,j,k}(t) = \frac{P(\mathbf{O}_r^{T_r}|\lambda_i)}{\sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|\lambda_l)} \gamma^r_{i,j,k}(t). \tag{4.10}$$

Since all derivatives are obtained, a remaining task is to determine the appropriate value of $D$ in Eq. 4.4. In this chapter, we follow Normandin's approach [32], which chooses the value of $D$ so that the mixture coefficient estimates remain positive, i.e.

$$D = \max_{c_{i,j,k}} \left\{ -\frac{\partial f_{MMI}}{\partial\, c_{i,j,k}}, 0 \right\} + \varepsilon, \tag{4.11}$$

where $\varepsilon$ is a small positive constant obtained from experiments.

### Gaussian Parameters

The re-estimation formula in Eq. 4.4 is applicable to the discrete output distributions. However, there is no closed-form solution for the EBW algorithm to be applied to the continuous densities. Normandin [32] suggests that the re-estimation formulae for means and covariance matrices of Gaussians can be obtained by considering a continuous Gaussian distribution as a limit of a discrete probability function.

Let $\mu_{k,d}$ and $\sigma^2_{k,d}$ be the $d$-th element of the means, $\boldsymbol{\mu}_k$, and the $d$-th variance of the diagonal covariance matrix, $\boldsymbol{\Sigma}_k$, respectively. The EBW formulae for Gaussian parameter updating [33, 34] can be adapted to the TDHMM context as follows [85]:

$$\hat{\mu}_{k,d} = \frac{(\Theta_{k,d} - \overline{\Theta}_{k,d}) + D\mu_{k,d}}{(\gamma_k - \overline{\gamma}_k) + D}, \tag{4.12}$$

and

$$\hat{\sigma}^2_{k,d} = \frac{(\Psi_{k,d} - \overline{\Psi}_{k,d}) + D(\sigma^2_{k,d} + \mu^2_{k,d})}{(\gamma_k - \overline{\gamma}_k) + D} - \hat{\mu}^2_{k,d}, \tag{4.13}$$

where

$$\Theta_{k,d} = \sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j \in Q_i^r, t=1}^{T_r} o_{r,t,d}\, \gamma^r_{i,j,k}(t)\, \delta(r,i), \tag{4.14}$$

$$\overline{\Theta}_{k,d} = \sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j \in Q_i^r, t=1}^{T_r} o_{r,t,d}\, \overline{\gamma}^r_{i,j,k}(t), \tag{4.15}$$

and

$$\Psi_{k,d} = \sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j \in Q_i^r, t=1}^{T_r} o^2_{r,t,d}\, \gamma^r_{i,j,k}(t)\, \delta(r,i), \tag{4.16}$$

$$\overline{\Psi}_{k,d} = \sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j \in Q_i^r, t=1}^{T_r} o^2_{r,t,d}\, \overline{\gamma}^r_{i,j,k}(t). \tag{4.17}$$

Because all states of all models share the same Gaussians, the notion of occupation counts is reduced to $\gamma_k$ and can be calculated by:

$$\gamma_k = \sum_{i=1}^{M} \sum_{j \in Q_i^r} \gamma_{i,j,k}, \tag{4.18}$$

where it is summed over all character classes and the states that the samples occupy. The anti-occupation counts, $\overline{\gamma}_k$, can be computed in a similar fashion. The value of $D$ is set to ensure that all variances are positive.

### 4.4.2 Gradient Descent Algorithm

Even though the derivatives of the MMI objective function with respect to various HMM parameters are given, the GD algorithm in Eq. 3.25 is not directly applicable to the optimisation of most parameters. The algorithm does not ensure that the resulting updates of mixture coefficients and transition probabilities remain positive and are subject to sum-to-one constraints. Similarly, the variances of the Gaussians are not guaranteed to be positive when they are optimised by the GD algorithm. To overcome these problems, a *softmax* transformation is introduced to enforce the positive-value and sum-to-one constraints. For example, when it is applied to mixture coefficient optimisation, the training process attempts to optimise the auxiliary function, which is a log function in this case, $h_{i,j,k} = \log c_{i,j,k}$ instead of optimising $c_{i,j,k}$, i.e.

$$\hat{h}_{i,j,k} = h_{i,j,k} + \eta \frac{\partial f_{MMI}}{\partial h_{i,j,k}}. \tag{4.19}$$

The softmax transformation is then applied:

$$\hat{c}_{i,j,k} = \frac{\exp \hat{h}_{i,j,k}}{\sum_{m=1}^{K} \exp \hat{h}_{i,j,m}}, \tag{4.20}$$

which ensures both the positive-value and sum-to-one properties, $\sum_{k=1}^{K} \hat{c}_{i,j,k} = 1$.

### 4.4.3 QuickProp Algorithm

A main disadvantage of the GD algorithm is its slow convergence rate and sensitivity to the learning rate. The use of higher-order derivatives has been introduced to improve the convergence rate. The second-order GD method, such as Newton's method, makes an explicit use of second-order derivatives. However, there is a limitation of its usability in practice because the actual Hessian matrix ($\nabla^2 f_{MMI}$) requires extensive computations. Therefore, systematic approximation is necessary. The QuickProp algorithm proposed by Fahlman [29] is an instance of the approximate second-order GD methods that has been widely used in neural networks-based systems and based on the diagonal approximation to the Hessian matrix. By applying the diagonal approximation to MMI training of the HMMs, it means that all HMM parameters are assumed to affect the value of the MMI objective function independently. The second derivative with respect to any model parameter $\lambda$ at current iteration $\tau$ is approximated by:

$$\frac{\partial^2 f_{MMI}}{\partial \lambda^2}(\tau) \approx \frac{\frac{\partial f_{MMI}}{\partial \lambda}(\tau) - \frac{\partial f_{MMI}}{\partial \lambda}(\tau - 1)}{\lambda(\tau) - \lambda(\tau - 1)}, \tag{4.21}$$

where $(\tau - 1)$ denotes parameters from previous iteration. The next iteration parameters are re-estimated by:

$$\lambda(\tau + 1) = \lambda(\tau) + \Delta\lambda(\tau), \tag{4.22}$$

which is the current parameter plus an updating step, $\Delta\lambda(\tau)$. The QuickProp algorithm involves using some heuristics to determine whether the approximated Hessian can be used for parameter updating. The updating step in Eq. 4.22 is computed differently, depending on the following conditions:

- If the current gradient has an opposite sign from the gradient of the previous iteration, the Hessian is used to compute the step size in a similar manner as in Newton's method: $\Delta\lambda(\tau) = \left[\frac{\partial^2 f_{MMI}}{\partial \lambda^2}(\tau)\right]^{-1} \frac{\partial f_{MMI}}{\partial \lambda}(\tau)$.

- If the current and previous gradients have a similar sign, an addition of a constant $\eta$ times the current gradient is required to ensure that the Hessian is properly conditioning and applicable for updating. In this case, the updating step becomes $\Delta\lambda(\tau) = \left[\frac{\partial^2 f_{MMI}}{\partial\lambda^2}(\tau)\right]^{-1} \frac{\partial f_{MMI}}{\partial\lambda}(\tau) + \eta \frac{\partial f_{MMI}}{\partial\lambda}(\tau)$, which is equivalent to the simple GD algorithm but with the help of the second-order derivative.
- Since the calculation of the Hessian is approximated, there is no guarantee that the approximated Hessian is valid, and the updating step may become too large. To cope with this situation, QuickProp limits the value of the updating step to the positive constant $\varphi$ times the updating step from the previous iteration, i.e. $\Delta\lambda(\tau) = \varphi\,\Delta\lambda(\tau-1)$.
- After computing the updating step, if it has an opposite sign from the current gradient, it implies that the search is in the wrong direction. In this case, QuickProp replaces the Hessian by zero and only the simple GD algorithm is used: $\Delta\lambda(\tau) = \eta \frac{\partial f_{MMI}}{\partial\lambda}(\tau)$.

### 4.4.4 Comparison of MMI Optimisation Algorithms

We conduct a number of experiments to evaluate the performance of various MMI optimisation algorithms discussed in the previous sections. The comparison focuses on the growth of the MMI objective function's value, $f_{MMI}$, over training iterations. The main purpose is to determine an effective optimisation algorithm to be used in the remaining experiments. The powerful algorithm should yield a reasonable improvement of $f_{MMI}$ within a small number of training iterations. Nevertheless, in MMI training where the objective function comprises thousands of parameters, a search surface is likely to be complex and requires a large number of training iterations in order to obtain a good solution [30].

The recogniser is operated in two stages: FM and DM. MMI training can be performed in either FM or DM training processes. However, the experiments in Sections 4.4.4, 4.5 and 4.6 focus on applying MMI training to the FM training process so as to optimise the FM models. The training and recognition processes are performed without having the DM stage involved, thereby reducing the processes in Figure 4.1 to Figure 4.3. The evaluation of MMI training in the DM training process is discussed in Section 4.7.

The evaluation of MMI optimisation algorithms is performed on a cut-down version of the database described in Section 4.2.1 to speed up comparisons. This would give a rough indication of the algorithms' effectiveness on the 5,000-sample database. A training set contains 1,000 samples (100 for each digit) randomly selected from such a database. Each training sample is represented by a sequence of 9-dimensional feature vectors or frames. The recognition performance of the testing set is not taken into account at this stage. We focus on the fully-tied TDHMM-based FM recogniser, and thus a codebook of 100 Gaussians is shared across all models. The codebook is initially generated by an unsupervised clustering ($k$-means algorithm, see Section 7.3.2) of all sample frames. The Gaussian parameters and mixture coefficients are optimised by ML training using

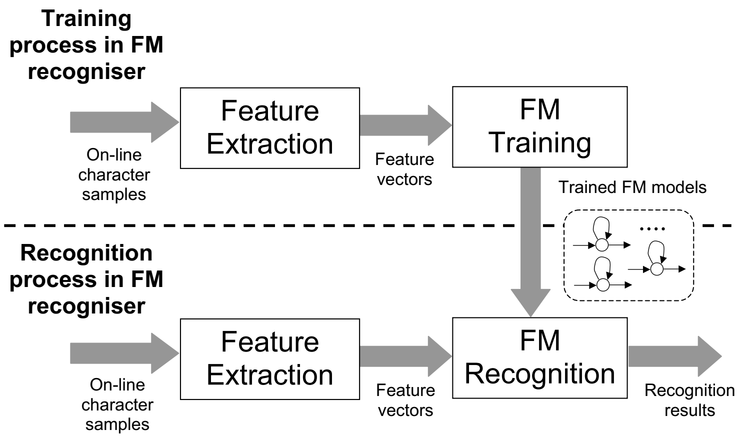


**Figure 4.3:** The training and recognition processes of the FM recogniser.

the Baum-Welch algorithm. Subsequently, MMI training is carried out for 40 iterations (an iteration refers to one pass over the whole set of training samples). The comparison of MMI optimisation algorithms focuses on optimising the mixture coefficients. The Gaussian parameters, which are not trained by MMI, are kept at their ML-estimated values.

The MMI optimisation algorithms involve some variables whose values are obtained from experiments. The variable $\varepsilon$ in Eq. 4.11 of the EBW algorithm is a small positive constant that controls a convergence rate. The performance of the EBW algorithm for various settings of $\varepsilon$ is presented in Figure 4.4 (a). It shows the growth of the MMI objective function over the number of MMI training iterations. It can be seen from Eq. 4.5 that the maximum value of $\frac{\partial f_{MMI}}{\partial c_{i,j,k}}$ is dominated by a small-valued parameter $c_{i,j,k}$. The smoothing constant $D$ can become very large, thereby making it insensitive to the small-valued $\varepsilon$. The experiments also show that setting $\varepsilon$ between 0.0001 and $0.1 \times \max_{c_{i,j,k}} \left\{ -\frac{\partial f_{MMI}}{\partial c_{i,j,k}}, 0 \right\}$ gives similar results and yields good convergence rates. Although the EBW algorithm is guaranteed to converge when the large value of $D$ (or $\varepsilon$) is used, such a value results in the convergence rate that is too slow to be useful (as seen in the graph when $\varepsilon$ is set to $10.0 \times \max_{c_{i,j,k}} \left\{ -\frac{\partial f_{MMI}}{\partial c_{i,j,k}}, 0 \right\}$).

As discussed in Section 3.5.2, Merialdo [69] argues that the search should place greater emphasis on high-valued parameters and replaces the derivative in Eq. 4.5 with the following formula:

$$\frac{\partial f_{MMI}}{\partial c_{i,j,k}} \approx \frac{\gamma_{i,j,k}}{\sum_{m=1}^{K} \gamma_{i,j,m}} - \frac{\overline{\gamma}_{i,j,k}}{\sum_{m=1}^{K} \overline{\gamma}_{i,j,m}}. \tag{4.23}$$

A different variant of the above equation based on the same idea is proposed by Normandin [32] as:

$$\frac{\partial f_{MMI}}{\partial c_{i,j,k}} \approx \frac{1}{c_{i,j,k}} \left(\gamma_{i,j,k} - \overline{\gamma}_{i,j,k}\right) \frac{\left(\gamma_{i,j,k} + \overline{\gamma}_{i,j,k}\right)}{\sum_{m=1}^{K}\left(\gamma_{i,j,m} + \overline{\gamma}_{i,j,m}\right)}. \tag{4.24}$$

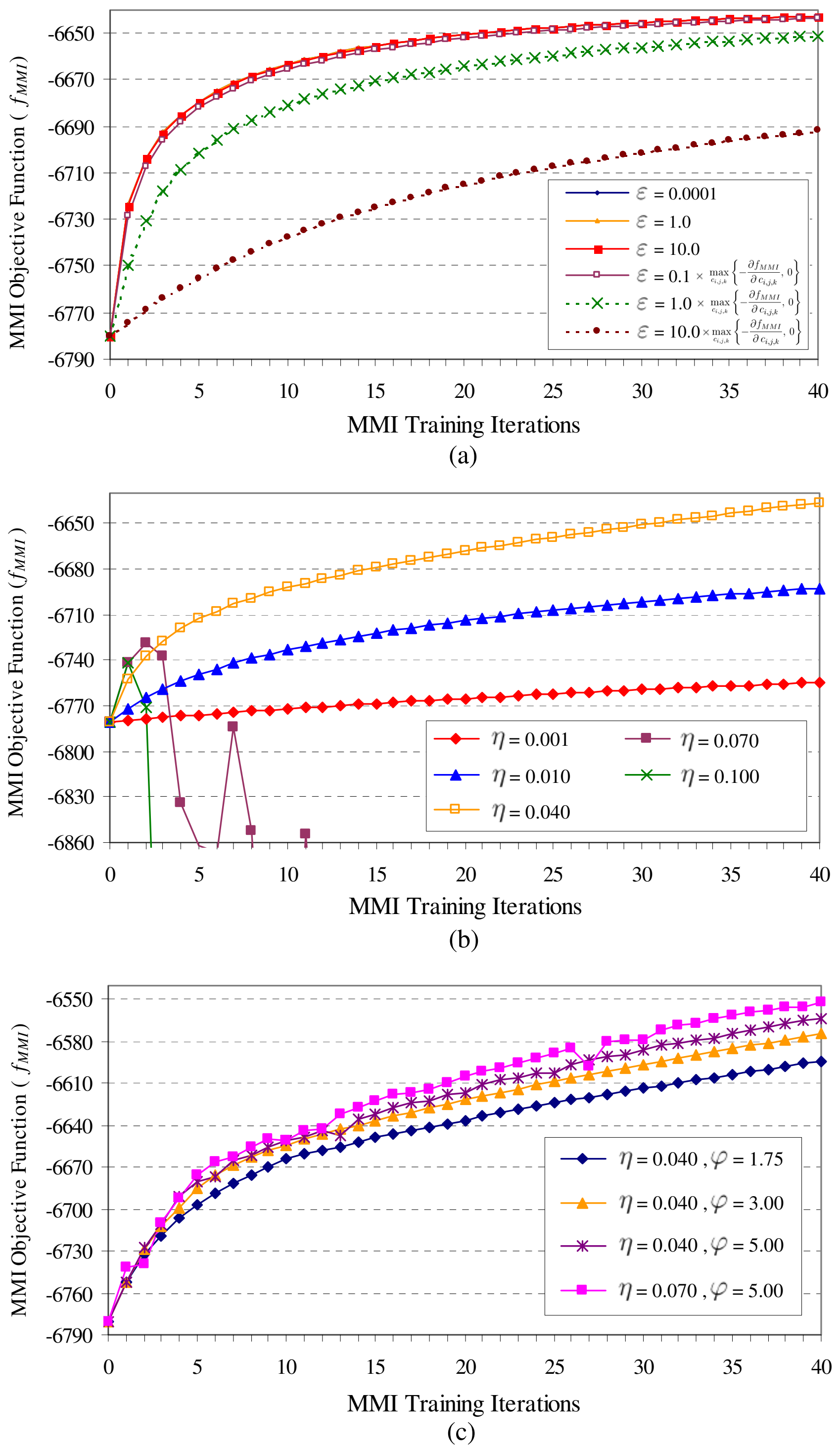


**Figure 4.4:** Evolution of the MMI objective function versus MMI training iterations for different settings of various MMI training algorithms: (a) Extended Baum-Welch (Note that setting $\varepsilon$ between 0.0001 and 10.0 gives similar results.), (b) Gradient Descent and (c) QuickProp

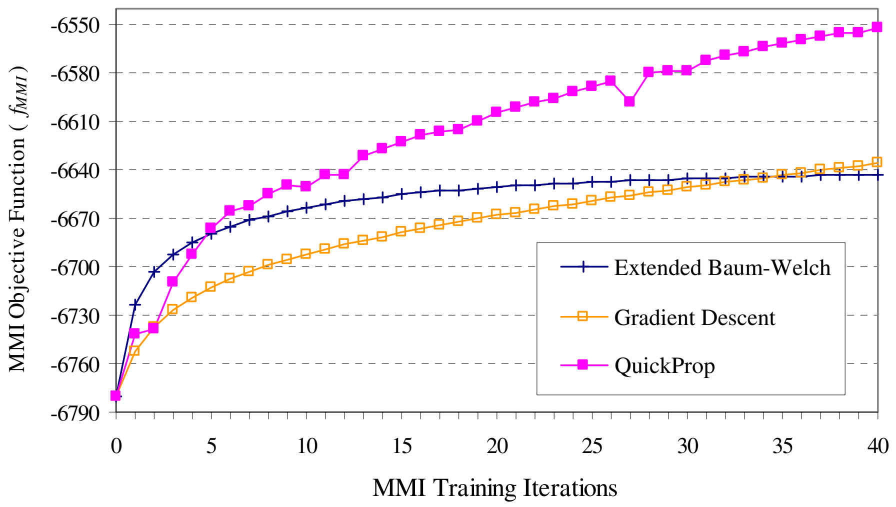


**Figure 4.5:** Comparison of the Extended Baum-Welch, Gradient Descent and QuickProp algorithms for mixture coefficient optimisation

These two alterations of the EBW algorithm have been evaluated in our experiments. Only a slightly better gain in $f_{MMI}$ is achieved when compared with the original EBW algorithm.

Figure 4.4 (b) illustrates the performance of the GD algorithm for various learning rates, $\eta$. Not surprisingly, the experimental results suggest that the GD algorithm is very sensitive to the setting of $\eta$. Setting an excessive learning rate results in a chaotic behaviour whereas a small learning rate yields too slow convergence rates. In contrast, the QuickProp algorithm is less sensitive to the setting of learning parameters than the GD algorithm, as shown in Figure 4.4 (c). The parameter $\varphi$ is chosen to be slightly larger than unity (a typical value suggested by Fahlman [29] is 1.75) to prevent the updating step becoming too large. From the experiments, a slight improvement in convergence rates can be accomplished by setting $\varphi$ to some larger values.

The best results of all three optimisation algorithms are compared in Figure 4.5. The EBW algorithm exhibits the highest growth rate during the first few iterations but it tails off quickly. In contrast, the QuickProp algorithm displays the highest growth of $f_{MMI}$ as the number of iterations increases. The evaluations of these MMI optimisation algorithms for Gaussian parameter training are carried out and also demonstrate a similar trend in that the QuickProp algorithm can give the highest performance. This may lead to a conclusion that, in our context, the QuickProp algorithm is best suited to MMI training. It is used as the algorithm for MMI training in the following sections.

## 4.5 MMI Training Performance and Parameter Discriminability

By referring to Section 4.4.4, MMI training is performed smoothly and increases the MMI objective function over training iterations. In this section, a series of experiments is conducted to evaluate the performance of MMI training by measuring the recognition accuracies of both training and testing sets. We continue to focus on the FM stage of the fully-tied TDHMM-based recogniser. A secondary goal for these experiments is to determine which model parameters play the most important role in discrimination. The parameters of the HMMs considered include: the mixture coefficients (MIX), the means of Gaussians (MNS) and covariance matrices of Gaussian distributions (COV). Prior to applying discriminative training, conventional ML training is used to initialise these HMM parameters. MMI training is then carried out by using the QuickProp optimisation algorithm. In the experiments, the parameters that are not re-estimated by MMI training are maintained at their ML-initialised values.

The experiments are performed on a full-version database, as described in Section 4.2.1, which consists of 5,000 training and 3,500 testing samples. The TDHMMs share a codebook of 160 Gaussians. Parameter updating in MMI training is carried out for 100 iterations. Figure 4.6 shows the evolution of the MMI objective function versus the number of iterations against various sets of MMI-trained parameters. The recognition performance of MMI training on the training set is depicted in Figure 4.7. The result of ML training is also shown.

As seen from these figures, mean and covariance optimisation is more effective than the mixture coefficient optimisation in terms of the growth of the MMI objective function. It also exhibits a higher gain of the recognition rates from the ML-trained result on the training set. The mixture coefficient optimisation yields only a small improvement of $f_{MMI}$. On the other hand, it results in a slight decrease in recognition rates. This is rather counter-intuitive in the context of the fully-tied TDHMMs because the mixture coefficients are the only parameters which are not shared across the models. They are therefore expected to carry more discriminative ability than the means and covariance matrices. The best result is achieved when the mixture coefficients, means and covariance matrices of Gaussians are optimised simultaneously. It steadily improves the recognition rate of the training set from 95.50% in ML training to 98.20% in MMI training.

The recognition performance of the testing set is shown in Figure 4.8. It demonstrates some fluctuations of the recognition rates over the training iterations. An improvement in the recognition results on the training set does not necessarily yield a similar result on the testing set. These results suggest that the *hold out method*, which utilises an independent validation set [12], should be employed to determine the termination of training iterations. Despite these fluctuating results, an analogous tendency to the training set's results is observed. That is to say, mean and covariance optimisation is more effective than mixture coefficient optimisation. The highest recognition result obtained over a series of MMI training iterations for each trained-parameter set is summarised in Table 4.1. Note that, by quoting the highest results in this way, they are likely to be biased as a performance measure and to indicate better performance than is likely on average. From the baseline ML-trained system, MMI training improves the recognition rate of the testing set from 95.40% to 96.29% when all HMM parameters are trained. This is equivalent to a 19% relative error reduction. The parameter that contributes the most is the covariance matrix whereas the means and mixture coefficients have slightly lesser contribution to discrimination.

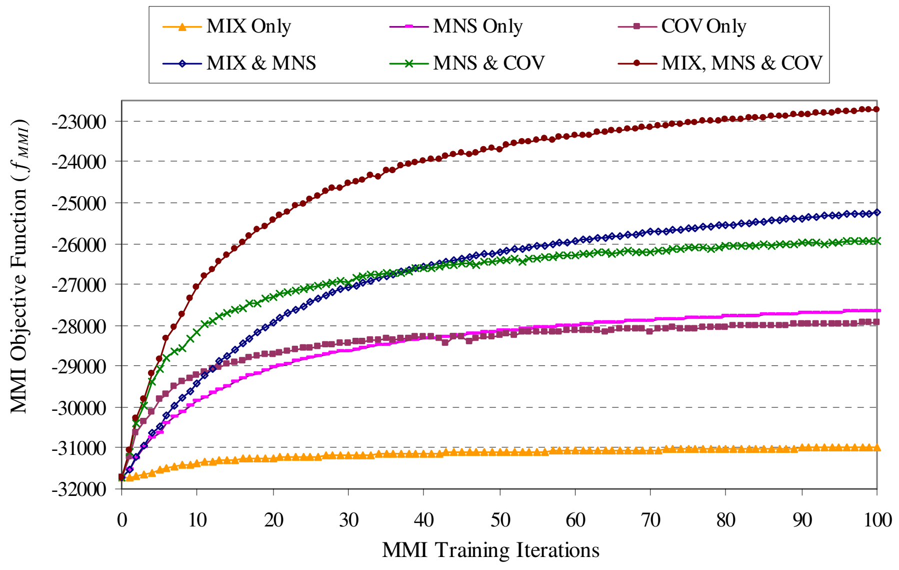


**Figure 4.6:** The growth of the MMI objective function versus the number of training iterations against different parameters optimised by MMI training in the fully-tied TDHMM-based system

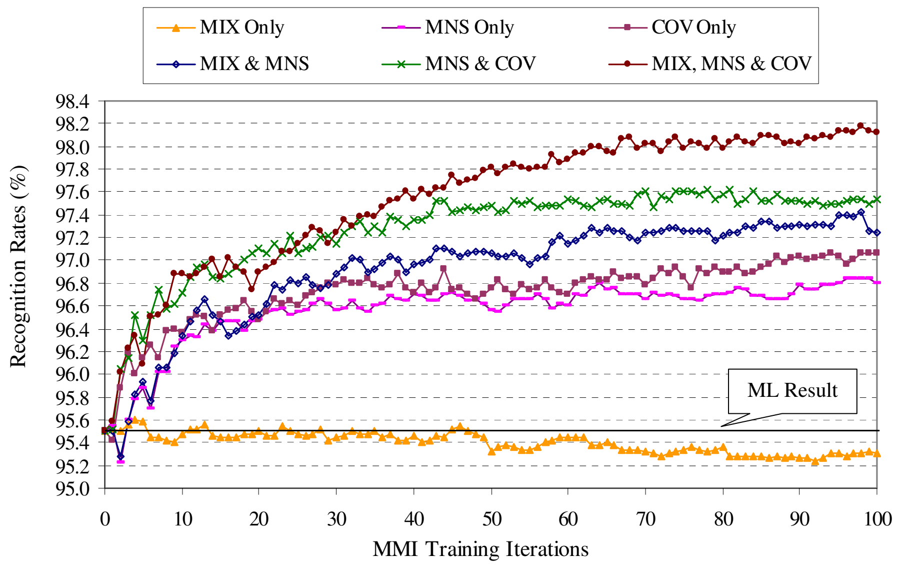


**Figure 4.7:** Recognition rates of the training set versus the number of training iterations against different parameters optimised by MMI training in the fully-tied TDHMM-based system

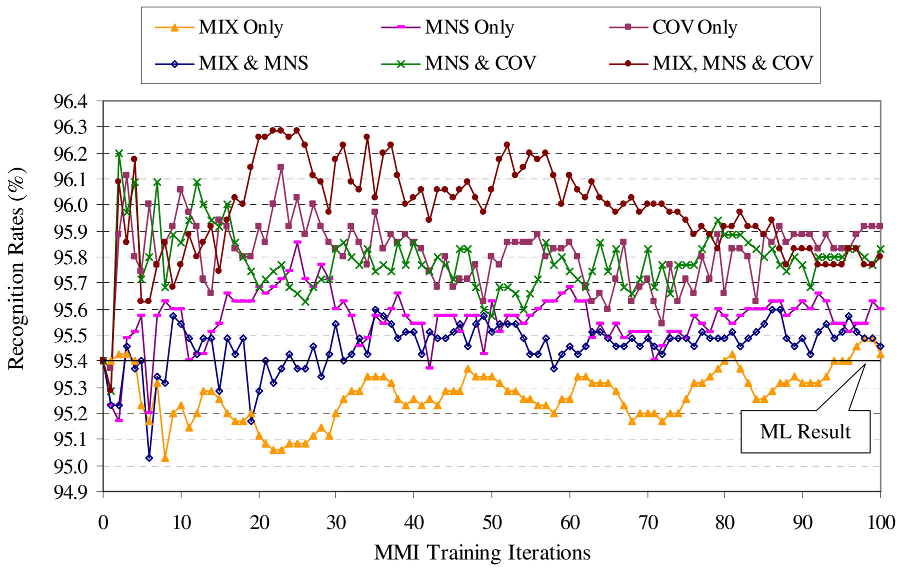


**Figure 4.8:** Recognition rates of the testing set versus the number of training iterations against different parameters optimised by MMI training in the fully-tied TDHMM-based system

| ML result | 95.40 % |
|---|---|

| Parameters optimised by MMI training | Recognition rates |
|---|---|
| MIX only | 95.49 % |
| MNS only | 95.86 % |
| COV only | 96.14 % |
| MIX & MNS | 95.60 % |
| MNS & COV | 96.20 % |
| MIX, MNS & COV | 96.29 % |

**Table 4.1:** Recognition results of MMI-trained fully-tied TDHMMs on the testing set compared with the result of ML training

Confusion matrices from the ML-training experiment and the best achievement of MMI training are presented in Table 4.2 (a) and (b), respectively. After MMI training, the number of correctly classified samples in most digit classes are increased, particularly the digit ‘6’ and ‘8’, while a slight increase of misclassified samples in some classes (e.g., the digit ‘0’ and ‘7’) is observed. In sum, it is shown that MMI training is effective despite being applied to such a highly-tied system as the fully-tied TDHMMs.

| Digit | Number of samples which are recognised as digit: | | | | | | | | | | Accuracies % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 0 | 324 | 0 | 7 | 1 | 0 | 0 | 4 | 0 | 13 | 1 | 92.57 |
| 1 | 0 | 328 | 3 | 1 | 1 | 0 | 0 | 17 | 0 | 0 | 93.71 |
| 2 | 0 | 1 | 339 | 2 | 1 | 4 | 0 | 3 | 0 | 0 | 96.86 |
| 3 | 0 | 0 | 5 | 340 | 0 | 4 | 0 | 1 | 0 | 0 | 97.14 |
| 4 | 2 | 6 | 0 | 0 | 337 | 0 | 0 | 1 | 0 | 4 | 96.29 |
| 5 | 0 | 0 | 1 | 2 | 0 | 345 | 0 | 0 | 2 | 0 | 98.57 |
| 6 | 24 | 0 | 0 | 0 | 1 | 5 | 318 | 0 | 0 | 2 | 90.86 |
| 7 | 0 | 4 | 1 | 0 | 0 | 2 | 0 | 341 | 0 | 2 | 97.43 |
| 8 | 20 | 0 | 0 | 1 | 0 | 1 | 2 | 0 | 326 | 0 | 93.14 |
| 9 | 4 | 0 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 341 | 97.43 |
| | | | | | | | | | Average | | 95.40 |

(a)

| Digit | Number of samples which are recognised as digit: | | | | | | | | | | Accuracies % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 0 | 321 | 0 | 1 | 1 | 0 | 1 | 4 | 0 | 20 | 2 | 91.71 |
| 1 | 0 | 328 | 0 | 0 | 4 | 0 | 0 | 14 | 0 | 4 | 93.71 |
| 2 | 0 | 0 | 344 | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 98.29 |
| 3 | 0 | 0 | 2 | 342 | 0 | 5 | 0 | 1 | 0 | 0 | 97.71 |
| 4 | 1 | 1 | 0 | 0 | 342 | 1 | 1 | 0 | 0 | 4 | 97.71 |
| 5 | 0 | 0 | 0 | 1 | 0 | 346 | 0 | 0 | 3 | 0 | 98.86 |
| 6 | 12 | 0 | 0 | 0 | 0 | 3 | 327 | 2 | 3 | 3 | 93.43 |
| 7 | 0 | 7 | 2 | 0 | 0 | 1 | 0 | 339 | 0 | 1 | 96.86 |
| 8 | 12 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 337 | 0 | 96.29 |
| 9 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 344 | 98.29 |
| | | | | | | | | | Average | | 96.29 |

(b)

**Table 4.2:** Confusion matrices for the fully-tied TDHMM-based system trained by (a) ML training and (b) MMI training

## 4.6 Effects of Parameter Tying in MMI Training

Even though some improvements from the ML training result are obtained in the previous section, the tied system may inhibit the actual discriminative ability of MMI training as argued in Section 4.3. In order to test this hypothesis, the same set of experiments are carried out on the CDHMM-based FM recogniser where there is no parameter tying involved. We use a total of 160 Gaussians, similar to the TDHMM-based system. However, each model has its own Gaussian pool with an equal number of Gaussians, 16 Gaussians per model in this case. The model parameters are initialised by ML training and MMI training is performed for 100 iterations by using the QuickProp optimisation.

The results in Figure 4.9 show the growth of $f_{MMI}$ versus the number of MMI training iterations for various parameters. We notice a similar trend as in the TDHMM case, that is, the optimisation of Gaussian parameters, especially the covariance matrices, results in a far greater improvement than the mixture coefficient optimisation. In terms of the recognition accuracies, corresponding results on the testing set are presented in Figure 4.10. The recognition results of the testing set are shown in Figure 4.11. These results are much more stable than those of MMI training in the fully-tied TDHMMs (see Figure 4.8). The best performance of the testing set obtained over a series of MMI training iterations is summarised in Table 4.3. Note that the baseline accuracy of the ML-trained CDHMMs is lower than that of the TDHMM-based system (93.71% in Table 4.1 compared with 95.40% in Table 4.3). The tied system uses a similar number of HMM parameters more efficiently than the untied system. By forming a pool of Gaussians, the number of Gaussian mixtures used for each state in the TDHMMs is much larger without increasing the total number of parameters to be estimated.

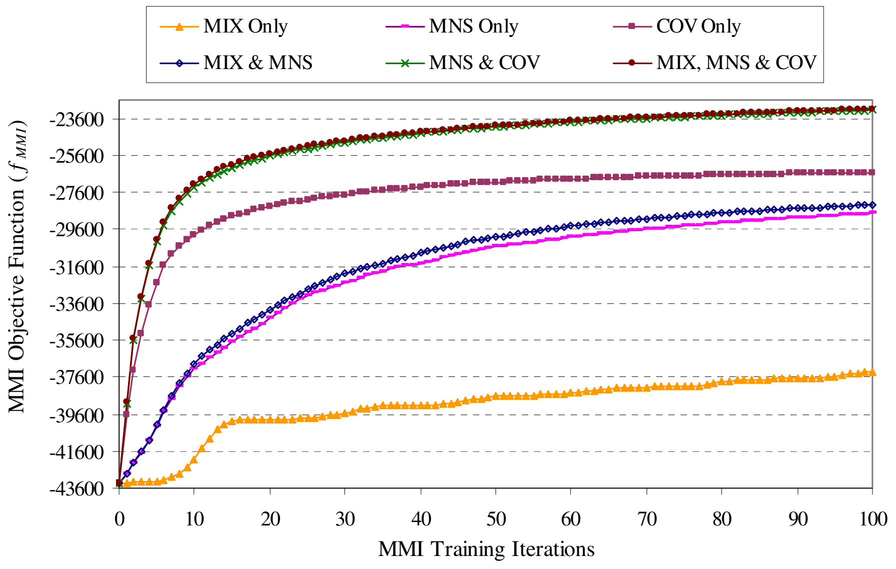


**Figure 4.9:** The growth of the MMI objective function versus the number of training iterations for different parameters optimised by MMI training in the CDHMM-based system

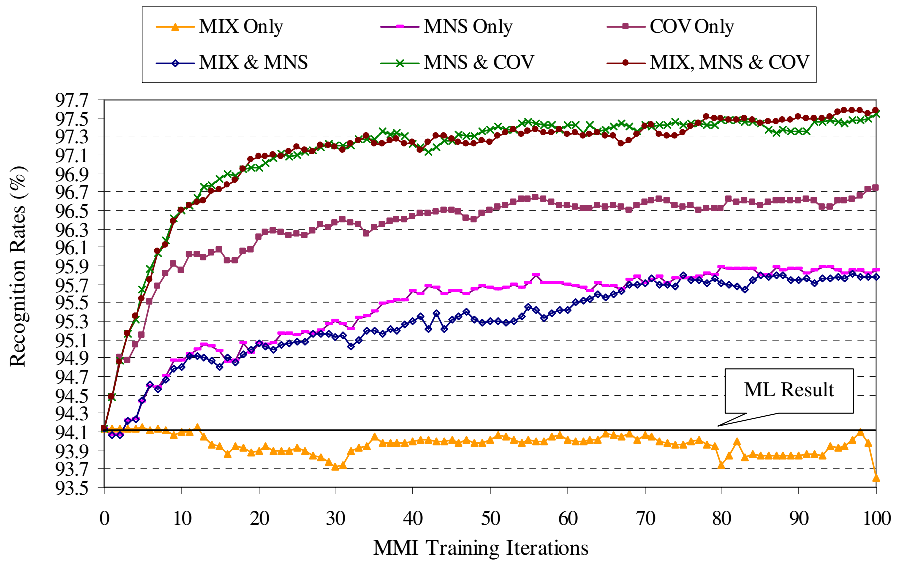


**Figure 4.10:** Recognition rates of the training set versus the number of training iterations for different parameters optimised by MMI training in the CDHMM-based system

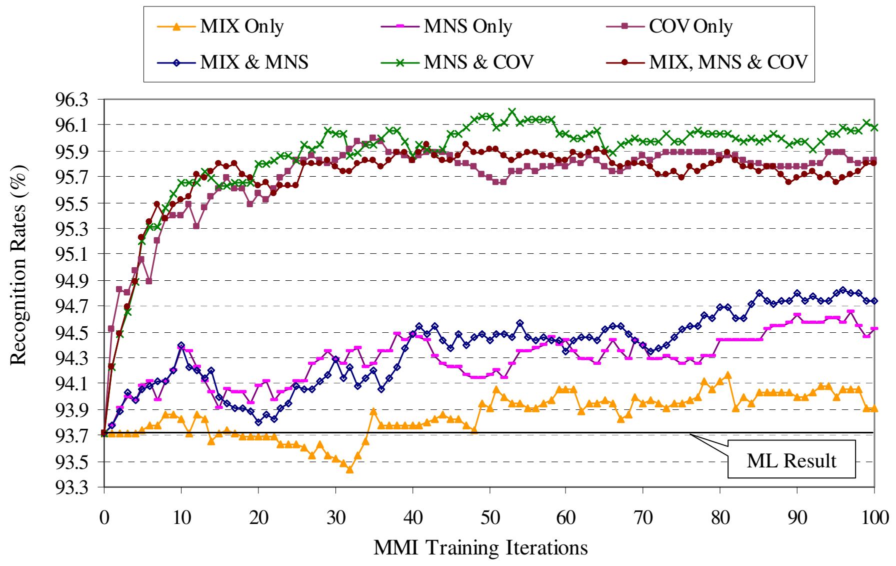


**Figure 4.11:** Recognition rates of the testing set versus the number of training iterations for different parameters optimised by MMI training in the CDHMM-based system

| ML result | 93.71 % |
|---|---|

| Parameters optimised by MMI training | Recognition rates |
|---|---|
| MIX only | 94.17 % |
| MNS only | 94.66 % |
| COV only | 96.00 % |
| MIX & MNS | 94.83 % |
| MNS & COV | 96.20 % |
| MIX, MNS & COV | 95.94 % |

**Table 4.3:** Recognition results of MMI-trained CDHMMs on the testing set compared with the result of ML training

From the results, the mixture coefficients have less discriminative ability than the means and covariances. The most discriminative parameters in this case are the covariance matrices of Gaussian distributions. The covariance matrix optimisation improves the result of the ML baseline from 93.71% to 96.00%, which is far better than the mean and mixture coefficient optimisation. The relative improvements from the ML-based system are much larger in the case of the CDHMMs than in the TDHMMs. The best result shows an improvement of the recognition rate from 93.71% in the ML-trained system to 96.20% in the MMI-trained system, or equivalent to a 40% relative error reduction. The confusion matrices for both systems are shown in Table 4.4. Clearly, MMI training is more effective when applied to the untied system.

## 4.7 MMI Training in Detailed-Match

The performance of MMI training in the previous sections has been examined in the context of the FM recogniser where single-state models are used. In this section, we evaluate MMI training performance by using the detailed-match (DM) recogniser. MMI training is applied to the DM training process (see Figure 4.1).

Each HMM in the DM recogniser is initialised from the corresponding ML-trained FM model by replicating the single-state model $N$ times. The number of states, $N$, in the DM model is chosen to be an average of the frame length of the training samples belonging to that model. The DM model that is initialised by the fully-tied FM model still preserves the parameter-tying property because only the state is replicated, but not its output distributions. In this case, the Gaussian pool is shared across all states of all models. This yields the fully-tied DM recogniser as illustrated in Figure 4.2 (c). A similar initialisation approach is also used to generate the DM model from the CDHMM-based FM model.[1] However, such an approach alters the behaviour of CDHMMs because all states of the resulting DM model essentially share the same Gaussian pool. This produces

[1]The output distributions of all states in the DM model are required to be identical due to the limitation of the recogniser structure.

| Digit | Number of samples which are recognised as digit: | | | | | | | | | | Accuracies % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 0 | 315 | 0 | 2 | 4 | 0 | 1 | 5 | 0 | 23 | 0 | 90.00 |
| 1 | 0 | 335 | 0 | 1 | 9 | 1 | 0 | 4 | 0 | 0 | 95.71 |
| 2 | 0 | 2 | 330 | 9 | 2 | 1 | 0 | 4 | 2 | 0 | 94.29 |
| 3 | 0 | 0 | 4 | 328 | 1 | 17 | 0 | 0 | 0 | 0 | 93.71 |
| 4 | 1 | 19 | 0 | 0 | 325 | 2 | 1 | 0 | 0 | 2 | 92.86 |
| 5 | 0 | 0 | 0 | 3 | 0 | 345 | 0 | 0 | 2 | 0 | 98.57 |
| 6 | 17 | 0 | 0 | 0 | 6 | 0 | 323 | 2 | 2 | 0 | 92.29 |
| 7 | 0 | 20 | 3 | 0 | 0 | 3 | 0 | 322 | 0 | 2 | 92.00 |
| 8 | 16 | 0 | 4 | 7 | 0 | 0 | 0 | 1 | 322 | 0 | 92.00 |
| 9 | 5 | 2 | 0 | 0 | 6 | 0 | 0 | 1 | 1 | 335 | 95.71 |
| | | | | | | | | | Average | | 93.71 |

(a)

| Digit | Number of samples which are recognised as digit: | | | | | | | | | | Accuracies % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 0 | 323 | 0 | 1 | 4 | 0 | 1 | 2 | 0 | 18 | 1 | 92.29 |
| 1 | 0 | 332 | 0 | 1 | 6 | 0 | 0 | 10 | 0 | 1 | 94.86 |
| 2 | 0 | 2 | 340 | 3 | 1 | 1 | 0 | 1 | 2 | 0 | 97.14 |
| 3 | 0 | 0 | 2 | 330 | 1 | 16 | 0 | 1 | 0 | 0 | 94.29 |
| 4 | 1 | 1 | 0 | 0 | 340 | 0 | 1 | 0 | 1 | 6 | 97.14 |
| 5 | 0 | 0 | 1 | 1 | 0 | 345 | 0 | 0 | 3 | 0 | 98.57 |
| 6 | 14 | 0 | 0 | 0 | 1 | 1 | 330 | 1 | 2 | 1 | 94.29 |
| 7 | 0 | 8 | 0 | 0 | 0 | 1 | 0 | 341 | 0 | 0 | 97.43 |
| 8 | 7 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 341 | 0 | 97.43 |
| 9 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 345 | 98.57 |
| | | | | | | | | | Average | | 96.20 |

(b)

**Table 4.4:** Confusion matrices for the CDHMM-based system trained by (a) ML training and (b) MMI training

the partially-tied DM recogniser where each model has its own set of Gaussians, as shown in Figure 4.2 (b).

After the DM models are initialised, ML training is applied to optimise the model parameters. MMI training is then carried out for 100 iterations by using the QuickProp algorithm. Thus, the MMI-trained DM models are achieved. The finalised FM models can be obtained from these DM models, as discussed in Section 4.2.2 (see Figure 4.1). These models are used in the subsequent FM and DM recognition processes. In the recognition process, given the testing sample, the FM recognition process generates a shortlist of the most-likely output candidates. It is found that nearly 100% of the correct recognition result falls within the top-five candidates of the list. These candidates are passed on to the DM recognition stage. The final recognition result is chosen as the model that yields the highest likelihood among those five DM models, rather than comparing all possible DM models.

The best result of MMI training in DM recognition is achieved when all parameters (MIX, MNS and COV) are trained. Figure 4.12 shows the recognition results of both the training and testing sets for the fully-tied TDHMM system versus the number of training iterations. The results of the partially-tied TDHMM system are shown in Figure 4.13. Fluctuations in the recognition rates can be observed in the fully-tied system whereas MMI training in the partially-tied system progresses much more smoothly. These results, together with the results in Figures 4.8 and 4.11, provide empirical evidence with the fact that tying parameters across the models causes an instability in MMI training. The recognition results of the testing set after 100 iterations of MMI training in DM recognition are summarised in Table 4.5, compared with the ML training results. It shows that MMI training is also effective in the DM recognition system.

## 4.8 Discussion and Summary

Discriminative training in the HMM-based recognisers has been applied to speech recognition problems effectively. However, its applications to the handwriting recognition problems are under-researched. In this chapter, MMI training is introduced as an alternative approach to ML training in the HMMs for the applications of on-line handwritten digit recognition. The results show some improvements in the recognition performance when compared to ML training. For example, in one experiment, MMI training increases the

| Types of DM recogniser | Recognition rates | |
|---|---|---|
| | ML training | MMI training |
| Fully-tied TDHMMs | 96.66 % | 96.94 % |
| Partially-tied TDHMMs | 96.23 % | 96.63 % |

**Table 4.5:** Recognition results of the testing set for the fully-tied and partially-tied DM recognisers achieved by ML and MMI training

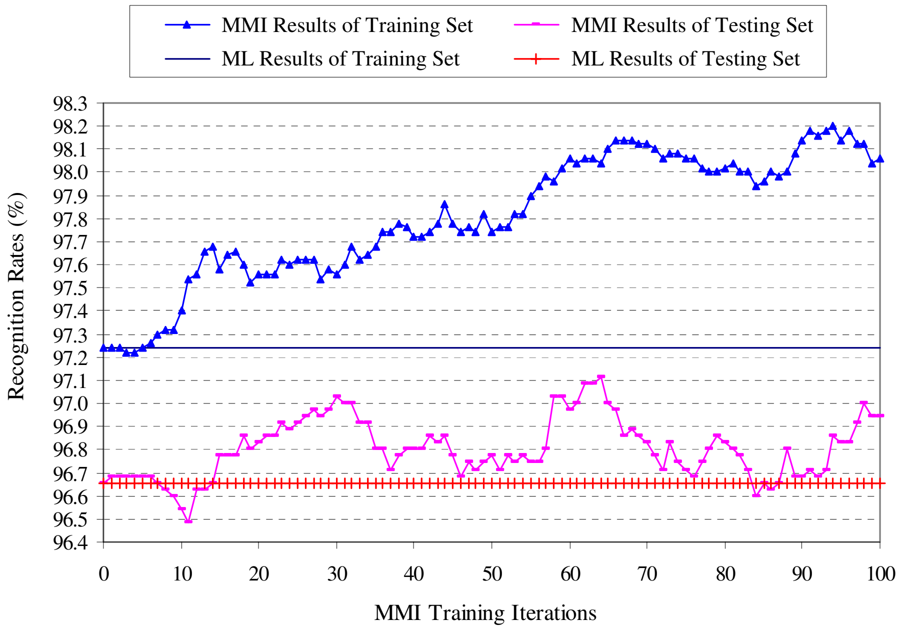


**Figure 4.12:** Recognition rates of the training and testing sets versus the number of MMI training iterations for the *fully-tied* DM recogniser compared with the result of ML training

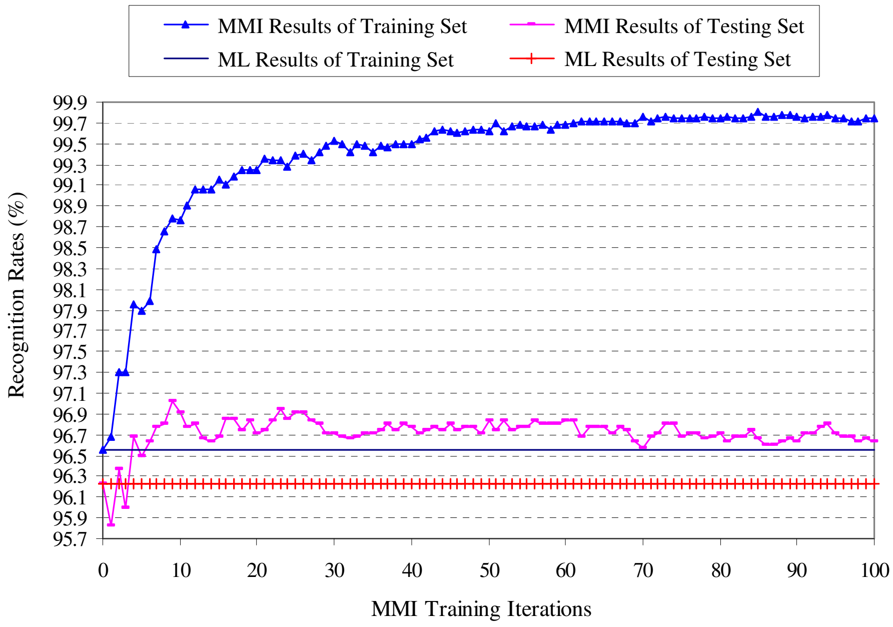


**Figure 4.13:** Recognition rates of the training and testing sets versus the number of MMI training iterations for the *partially-tied* DM recogniser compared with the result of ML training

recognition rates of the testing set from 93.71% on the ML-trained system to 96.20% on the MMI-trained system, or equivalent to a 40% relative error reduction. Several aspects of MMI training in the HMMs are investigated. A number of contributions of the present study can be summarised as follows:

- MMI training is effective despite being applied to the highly-tied system such as the fully-tied TDHMM-based recogniser. The structure of this recogniser is different from most applications of MMI training in the literature. That is to say, the Gaussian parameters in the system are shared among all states of all HMMs. Optimising the Gaussian parameters of one model has an explicit impact on those of other models, and thus it may inhibit the discriminative ability of MMI training. The best achievement improves the recognition rates of the testing set from 95.40% on the ML-trained system to 96.29% when MMI training is used. This is equivalent to a 19% relative error reduction.
- Three different training algorithms for MMI training in the context of TDHMMs, including the Extended Baum-Welch, Gradient Descent and QuickProp algorithms, are evaluated. In terms of the growth of the MMI objective function over the training iterations, the gradient descent algorithm yields the slowest convergence rate among the three algorithms. Although the Extended Baum-Welch gives the highest growth rate during the first few iterations, the QuickProp algorithm is more efficient in the long run.
- A number of studies to determine which MMI-optimised model parameters play the most important role in discrimination are carried out. It is shown that the optimisation of Gaussian parameters is more effective and exhibits higher gain in recognition rates on the testing set than the mixture coefficient optimisation. This is rather counter-intuitive in the context of fully-tied TDHMMs because the mixture coefficients are the only parameters which are not shared across the models and therefore expected to carry more discriminative ability than the means and covariance matrices.
- In order to test the aforementioned hypothesis as to whether a highly-tied environment obstructs the actual performance of MMI training, the same sets of experiments are applied to the untied system based on the CDHMMs, where there is no parameter tying involved. It shows that MMI training is more effective when applied to the untied system with the same number of Gaussian parameters. That is, a 40% relative error reduction compared to ML results is obtained whereas only a 19% relative error reduction is achieved in the fully-tied TDHMM case. Experimental results confirm that the optimisation of Gaussian parameters, the covariance matrices in particular, provides the maximum discriminability in MMI training.
- The performance of MMI training on the fully-tied system is compared with that on the partially-tied TDHMMs where the Gaussian parameters are shared among the states only within the same model, but not across the other models. Results of MMI training on the fully-tied system demonstrate that the recognition rates of the testing set fluctuate over the course of MMI training. On the contrary, MMI training on the partially-tied system and the untied system progresses much more smoothly. The results indicate that tying parameters across the models causes an

uncertainty in MMI training, and the *hold out method* which utilises an independent validation set should be used to determine the termination of training iterations.

The experimental results in this chapter reinforce that MMI training offers a consistent improvement in the recognition performance over ML training when applied to the on-line handwritten digit recognition task. MMI training is, therefore, regarded as a potential alternative approach to ML training for larger-scale handwriting recognition problems. Chapter 5 introduces the problem of Thai character recognition, and the applications of MMI training to this problem are extensively studied in Chapter 6.

The gains in the recognition accuracies from ML training are obtained at the expense of additional computations required in MMI training. Convergence and computational efforts involved are the most important factors to be considered when choosing the optimisation algorithms. Although the QuickProp algorithm is selected for MMI training in this chapter, we experience some difficulties when it is applied to the multi-state HMMs. That is to say, the more the number of model parameters grows, the more sensitive the algorithm becomes towards the settings of the learning rate, $\eta$, and $\varphi$ variables for each optimised parameter. A variation of the Extended Baum-Welch algorithm, which is less sensitive to the smoothing-parameter setting, proposed by Povey [10] is investigated in Chapter 6.

# Chapter 5

# Thai Character Recognition

## 5.1 Introduction

This chapter gives an overview of the problem of Thai character recognition, both machine-printed and handwritten characters, as background knowledge for the applications of discriminative training and HMMs to this problem in Chapter 6. To be able to solve the problem efficiently, knowledge of the Thai language is an inevitable requirement, or at least to a certain level. It is worth presenting the basic concepts of the Thai language and its writing styles in Section 5.2, followed by a survey and review of the related literature in Thai character recognition in Section 5.3. The aim of this review is to describe the previous research reports of Thai character recognition and to point out their common constraints which the techniques presented in Chapter 6 are proposed to overcome. The summary of the chapter is presented in Section 5.4.

## 5.2 Characteristics of the Thai Language

In Thailand, Thai serves as the official national language. It is the language used in schools, the media and government affairs. More than ninety percent of its population speak and write Thai as their native language. The introduction to the Thai language in this section aims to illustrate the scripts used to write in the Thai language and its writing style, which is unique and well understood throughout the country. This is sufficient as a background to understand the nature of the problem of classifying Thai characters. It does not intend to give either the detailed background of the history of the Thai language nor its grammar. In brief, the recorded history of the creation of the Thai writing system dates back to the mid-thirteenth century. It was being constantly modified under the influence of the Khmere, Sanskrit and Pali languages until the late-eighteenth century. More details of Thai language history can be found in [86, 87].

Thai script is written from left to right, with lines going from top to bottom of a

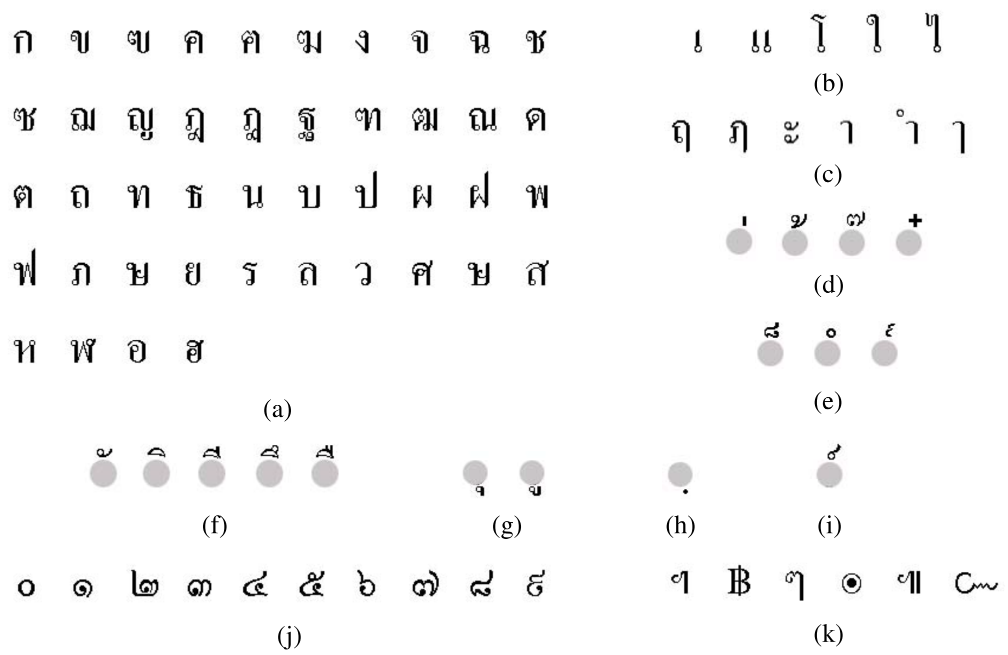


**Figure 5.1:** The Thai alphabet set and its categories, (a) consonants, (b) lead vowels, (c) follow vowels, (d) tone marks, (e) above diacritic 1, (f) above vowels, (g) below vowels, (h) below diacritic 2, (j) numerals, (k) symbols; Note that the purpose of the *grey circle* sign is to show the level of each character. When writing Thai, the grey circle is replaced by one of the consonants.

page. The standard Thai writing style is in block letters, and is not cursive as it is in the Arabic and many English writing styles. This means each Thai character is well separated from its neighbours by a gap. However, Thai words can be written contiguously, without extra space between words, as opposed to separating each word with space as in English. Using extra space in Thai writing style indicates only the beginning of a new phrase or sentence.

The Thai alphabet consists of 87 characters, which include consonants, vowels, numerals, tone marks, diacritics and symbols. Among these characters, ten characters are obsolete and no longer used in common writing. All characters are categorised and shown in Figure 5.1. The consonants are written on the baseline. However, as in English, some consonants have an ascender or descender which also covers the area above or below the baseline. In addition, some characters (tone marks and above vowels) are written above other characters while some are placed under the baseline. This forms a four-level writing structure. For example, the word in Figure 5.2 consists of seven characters written in four different levels.

The Thai language offers a unique challenge to automatic handwriting recognition on the grounds that there are several groups of Thai characters which have similar shapes, and vary only in small details such as loops and curls (for example, the second and third consonants on the top line of Figure 5.1 (a)). These details are well preserved in the printed Thai characters as shown in Figure 5.1. However, they are often neglected in

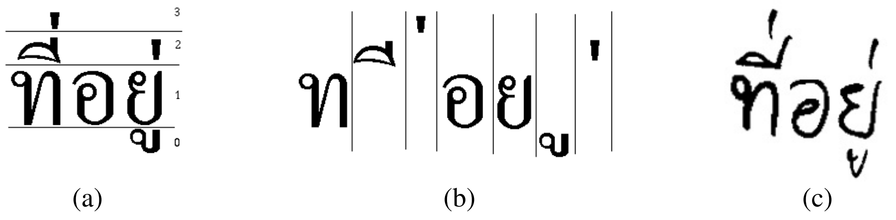


**Figure 5.2:** Printed and handwritten versions of a Thai word, *teeh-yooh* 'address', showing four-level writing style, (a) a correct printed version, (b) an incorrect printed version of the same word showing each character in different cursor positions and (c) a handwritten version of the same word

several handwriting styles. Thai handwriting styles can be broadly categorised into two groups: the *constrained* and the *unconstrained* styles. The constrained or handprinted style is what children learn at schools from the first grade. This style is intended to be analogous to the printed Thai characters. Each character is written neatly showing the distinct features of the characters clearly. On the contrary, with an unconstrained-writing style, people tend to write less carefully and omit those important details. Loops are not fully closed or are written as blobs. Curls are likewise elided or attenuated. Figure 5.3 shows samples of the constrained and unconstrained Thai handwriting styles. When writing is performed at high speed such as in note-taking, it usually brings out an additional difficulty of touching characters which is beyond the scope of this work. We focus on the recognition of the unconstrained-style handwritten Thai characters without any touching characters involved, or in other words, off-line isolated Thai handwriting recognition. This research also deals with variations in handwriting styles from multiple writers. Form filling and formal letter writing are examples of this writing style. The 'unsolved' problem of recognising unconstrained-style Thai handwriting with touching characters relies on, to some extent, the correctness of isolated character recognition. Hence, this research is regarded as an essential foundation for the applications of isolated character recognition to such a considerably more difficult problem (see Section 9.2.2).

## 5.3 Previous Research in Thai Character Recognition

The use of computers to store and manipulate Thai characters started in 1967 with the development of a Thai character code for IBM punched cards and line printers. The earlier applications were limited to mainframe data processing tasks until 1982 when the first Thai text editor package, *Thai Easy Writer*, was released. Since then, several microcomputer Thai applications have been developed and widely used in both business and research areas, but without a common standard for Thai character encoding. The standard for Thai character codes was released in 1986 by TISI (Thai Industrial Standards Institute) and revised by them in 1990. This standard is known as TIS 620-2533.[1]

[1]http://www.nectec.or.th/it-standards/mlit97/keynote.html.

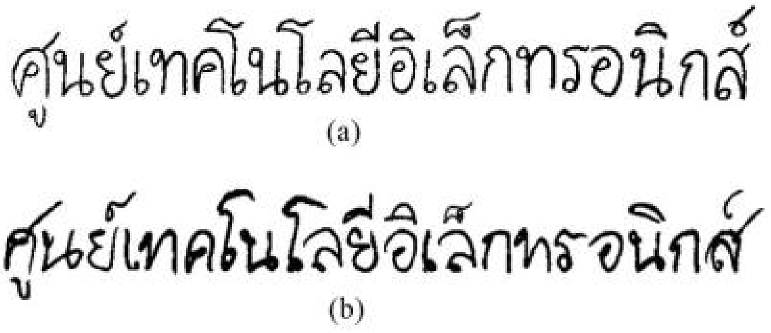


**Figure 5.3:** Samples of the handwritten Thai phrase showing different writing styles, (a) constrained-style writing, (b) unconstrained-style writing.

The development of Thai character recognition has progressed slowly as can be seen from a small number of published reports and papers over the past twenty years. The main factor in the delay is the late start in the research. Research in Thai character recognition emerged in 1982, compared to Latin in the 1950s and Chinese and Japanese in the 1960s. Most of the work is in the area of machine-printed Thai character recognition while only a small number of reports address the problem of handwritten Thai character recognition. Two commercial products for Thai Optical Character Recognition (OCR) for printed documents first appeared on the market in 1996 and 1997.[2] They are claimed to achieve an accuracy of 90% to 95% in recognising fixed fonts. None of the published work shares the same dataset set or samples used for evaluation. Hence, the results are not practically comparable, and it is suggested that there should be a common database to facilitate research in the field. In the past few years, the number of published papers in the field of handwriting recognition have significantly increased due to more interest in the field and pressure from the market, which can be seen as a promising signal for the advance of research and co-operation among researchers.

The following sections review previous research in Thai character recognition. The review is categorised into several fields of Thai character recognition research. These include pre-processing techniques (Section 5.3.1), off-line printed Thai character recognition (Section 5.3.2), off-line handwritten Thai character recognition (Section 5.3.3), and on-line Thai handwriting recognition (Section 5.3.4).

### 5.3.1 Pre-Processing Techniques

In off-line character recognition, the first step is to acquire a text document and transform it into a scanned image. The off-line recognition system recognises the text after it has been written or typed. Usually, the raw data of the scanned text image cannot be processed directly. Pre-processing techniques are required in order to convert the image into a suitable format for the recognition process. The result of the pre-processing stage should be an image containing the words or characters to be recognised without any disturbing elements. Examples of pre-processing techniques include binarisation or thresholding, noise reduction or smoothing, document skew detection and correction,

[2]The software ARNThai by NECTEC, and the Atrium ThaiOCR by Atrium Technology.

document decomposition and slant normalisation. In addition, the processes of thinning, skeletonisation and segmentation [88] are sometimes embodied in the pre-processing stage. There are a number of research reports which introduce novel methods to improve the performance of the pre-processing techniques in Thai character recognition. A review of these techniques is given below.

Often, the document to be recognised is not placed correctly on the scanner. For example, it may be skewed on the scanner bed which results in a skewed image. The skewed document has a disadvantageous effect on character segmentation and recognition. The automatic skew angle detection and correction processes should be done prior to the recognition process in order to attain optimal performance. Duangphasuk and Premchaiswadi [89] propose a method to detect and correct skew angle in the document by using a linear regression algorithm. This method is claimed to be faster than the conventional Hough transform algorithm [90] while maintaining comparable accuracy. On the other hand, document skew correction can also be done manually, as described in [91] where the users are asked to manually correct the skew angle.

The problem of slant characters or words may be occurred in some handwriting styles or the italic fonts used in machine-printed text. Slant angle estimation and slant normalisation are required to normalise these slanted characters. The slant normalisation algorithm in [92] estimates the slant angle from the lines connecting a centre of gravity of each horizontal strip for each individual character. Then, an affine transformation is applied to all pixels to normalise the slant characters. The method based on Variation Entropy [93] is applied to the slant normalisation problem of Thai characters. Note that the specific knowledge of the Thai language is not a requirement to solve the problem of skew and slant detection. Therefore, the common techniques used in other languages can be applied to this problem without further modification.

The thresholding process, which is required to convert the scanned images into bi-level images, may unnecessarily yield a significant loss of information and subsequently result in broken parts in the characters. Recently, a mending algorithm for broken printed Thai characters has been proposed [94]. The information of the overlapping areas of the broken image, the specific features of the character and its projection profile are employed in this repairing process. Understanding of Thai writing characteristics is required to solve this problem efficiently.

One of the most important problems that affects the accuracy of character recognition systems for Thai language is the segmentation of the Thai characters. The classical method for segmenting Thai characters from a sentence is to use their vertical and horizontal profiles [95, 96]. Because the construction of Thai words is different from that of English words, only the projection of the vertical profiles cannot separate the Thai characters. An alternative approach is to use the average value of the horizontal projection profiles to obtain a consonant line level, and a contour tracing algorithm to restore the deleted part [92, 97]. Santinanalert [91] uses connected component analysis for Thai character segmentation. Figure 5.4 shows an example of segmenting a handwritten Thai phrase into individual characters using connected component analysis. The problem of segmentation seems to be more serious when two characters are touching or crossing one another. Unlike other languages, touching and crossing Thai characters can occur both

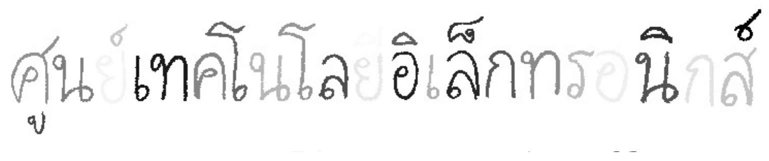


**Figure 5.4:** Connected component analysis of a Thai phrase. Each connected component is indicated by a distinct grey level.

horizontally and vertically while in other languages they usually occur only horizontally. A new segmentation scheme for touching Thai printed characters is proposed by Premchaiswadi *et al.* [98]. In [98], a combination of an edge detection algorithm, horizontal and vertical projection profiles and a statistical width and height of the characters is used to determine where to split the touching characters. On the other hand, the crossing character segmentation techniques in [99, 100] require both finding the crossing point and reconstructing the segmented characters. Segmentation of touching Thai characters is considered to be a difficult problem, and it requires detailed knowledge of the Thai writing system. The segmentation problem remains unsolved, especially in a more difficult case of unconstrained Thai handwriting. However, this problem is beyond the scope of this research since we only focus on the recognition of isolated handwritten characters.

### 5.3.2 Printed Thai Character Recognition

This section reviews some of the off-line printed Thai character recognition systems which have been detailed in print.

Most of the earlier works use template-based recognition systems. Since there are many similar characters in Thai, Kimpan *et al.* [95] separate their recognition process into two stages: *rough* and *fine* recognition. In the rough recognition stage, similarity coefficients among the characters are measured as a criterion for categorising them into several groups. The eigenvalues and eigenvectors of the Karhunen-Loève expansion (K-L expansion) of each group are computed. The eigenvectors having maximum eigenvalues, which constitute the principal components in each category, are used as a template of that group for matching with unknown characters. In the fine classification stage, explicit pieces of the character, which show the important characteristics of that character, are used as the standard patterns for matching with the same position pieces of unknown characters. An improvement in the fine classification stage is presented in their later work [101] where linear decision functions of the distributions in an eigenvector space are constructed to discriminate single-font printed Thai characters in each category. The explicit pieces of the character are still effectively used in the process of discriminating very similar characters. This recognition process has demonstrated its success in recognising the single-font and single-sized printed Thai characters. The same method, which separates the recognition process into rough and fine recognition and uses explicit pieces of the character to form templates, has been applied to the work of Srisuk [102]. In his work, the Hausdorff distance is used to measure the similarities among characters for both rough and fine classification.

The disadvantage of these classification methods that employ two classification stages (rough and fine classification) is that the recognition accuracy relies on the performance of both stages. If the character is misclassified in the first stage, it is impossible that correct final recognition can be achieved in the second stage.

A recognition method for printed Thai characters by structural analysis of their contours has been proposed by Hiranvanichakorn *et al.* [103] where digital contours of the characters are encoded according to their directional differences. Arithmetic operations are then applied to extract concavities and convexities of the contours. In their recognition process, several geometric features of concave and convex arcs are used to calculate similarities of the arcs. Recognition of an unknown character is made by detecting similar arc pairs. A modification of this method is reported in [104] where a more effective method to extract the concavities, convexities and features for the recognition of low-resolution Thai characters is proposed. A freeman chain code and the directional differences of contour tracing of the characters are utilised to extract concavities and convexities. However, detecting similar arc pairs by this method is rather complex. It is difficult to obtain an optimum result in recognition, and it also takes a long computation time for matching.

The work in [105] uses topological properties for both rough and fine classification. A thinning algorithm is applied to make it easier to extract topological properties. The topological properties of characters are represented in the form of end strokes, intersection strokes, independent or inner strokes, and these are used to discriminate between characters. A decision table for each category based on their topological properties has been built to classify each character in the subgroup. However, this method is difficult to apply to some of the similar characters. In addition, the decision table has to be designed manually, which is a very difficult and time-consuming task.

Another attempt is based on inductive logic programming (ILP) and neural networks [106, 107]. ILP is able to employ domain-specific background knowledge that provides better generalisation performance on data. Structural features are extracted from the thinned character and are represented as a primitive vector list where each primitive vector represents the types of the lines or circles in the character. The output after training is a set of learned ILP rules. Each rule defines the characteristics of one character. A backpropagation neural network is employed for the approximation of ILP rules which are later used to match with unseen or noisy characters. In [91], feed-forward neural networks with backpropagation learning are also used to classify printed Thai characters, using feature vectors based on eigenvectors and a K-L transform of the character images.

The work by Marang *et al.* [96] combines several techniques, such as boundary normalisation (elliptic Fourier series), fuzzy set theory and neural networks, to recognise printed Thai characters. Boundary functions of characters are detected and normalised so that they are invariant to orientation and dilation of the characters. A fuzzy membership function is constructed for mapping values of these functions from different discrete points into fuzzy values. Then, a feed-forward neural networks with a backpropagation learning algorithm is used to learn fuzzy inference rules. Alternatively, Kawtrakul *et al.* [108] propose a recognition system based on multiple feature extraction and a minimum distance classifier. These features include the direction of the character contour, the density of the character body and the character's *peripheral information* (i.e. the distance from the normalised image frame to the nearest contour). Although the results obtained by these techniques achieved acceptable recognition rates, they are evaluated on printed Thai characters with only few fixed fonts and sizes.

### 5.3.3 Handwritten Thai Character Recognition

In the late 1960s, handwritten character recognition was put into practical use in USA and Japan for postcode reader applications. Over the past 25 years, the field of character recognition has made significant advances in handwriting recognition [109]. However, the performance of handwriting recognition systems, especially for the off-line unconstrained-writing style, is still inadequate for practical applications. The problem involves high variabilities of handwriting styles. A number of researchers in both academic and industrial sectors are actively working on this subject. Although the first attempts at Thai handwriting recognition were in the mid-1980s, it is only recently that there has been significant research activity. The published literature on Thai handwriting recognition can be surveyed almost in its entirety as follows.

Hiranvanichakorn *et al.* [110] introduce a recognition method for handprinted Thai characters based on concavities and convexities of contours, similar to their previous approach [104] described in Section 5.3.2. The additions to that approach are the dictionaries of the characters and the arcs which are subsequently used to produce a compact and effective dictionary of the models to be used in the matching process. In contrast, Airphaiboon and Kondo [92] use the loop structure and local features of the character in their recognition system. These properties include the number and locations of *end points* and the loop structure of the processed character image. The rough classification stage is used to divide all characters into several categories. In the recognition process, a decision tree is manually designed to classify the characters in each category. Additional features based on small details in the character are also applied to the decision process for some character categories. Phokharatkul and Kimpan [97] applies the same approach as in [92] for rough classification. However, the fine classification stage in [97] uses Fourier descriptors which describe the closed contour of the character as the features to train neural networks. A combination of a genetic algorithm (GAs) and a backpropagation learning algorithm is then applied to compute the optimal weights of the neural networks. The common disadvantage of the approaches in [92, 97, 110] is that they are not applicable to an unconstrained writing style where the loops in the characters are often not closed or are written as blobs. Discontinuity in character images is also a problem.

Veerathanabutr and Homma [93] propose an alternative method that extracts features of character images from their contour direction codes by using the Fuzzy Direction Code Histogram (FDCH). Other details of the character, such as the number of loops, its width and height, are also used to form the feature vectors. Feed-forward neural networks are employed in their recognition process. In contrast, Methasate *et al.* [111] introduce a feature extraction method that extracts global and local features of the character. The global feature is the width to height ratio of the character whereas the local features are the relative positions of all loops, end points or curls of the character. Membership functions are then applied to map these features into many fuzzy terms required by the fuzzy-based classifier. The fuzzy-syntactic decision method that employs fuzzy inference rules is subsequently used to recognise each character. The method in [112] also uses distinctive features of the Thai characters, namely loops, curls and stroke changing, for segmenting the strokes in Thai handwriting.

Having considered the detail of the aforementioned techniques, it can be seen that they rely on the presence of important features of Thai characters, such as loops and curls, which have to be clearly written. However these features are often omitted or less visible in the unconstrained-writing styles even in the case that the handwritten characters are well separated. The requirement that these constraints have to be conserved to successfully apply these techniques makes them less applicable to the actual problem. A recent work by Chatwiriya [113] proposes a method for off-line handwriting recognition of the Thai currency amount written on bank cheques that uses a combination of multiple features which does not rely on the distinctive features of the Thai characters. Such a recognition problem is narrowed down to be the recognition of only 25 Thai characters that can form 17 words used to write the currency amount. Although the recognition accuracy of individual characters is not very high, the overall accuracy of the system is very good due to the use of post-processing techniques and a constrained small number of vocabularies.

### 5.3.4 On-line Thai Handwriting Recognition

In on-line handwriting recognition, characters are recognised while they are being written on an appropriate device, such as an electronic tablet or a combination of screen and digitiser that instantaneously displays what the user writes. Dynamic information captured during a writing process includes trajectories, duration and the order of the strokes.

There have been only a few attempts at on-line Thai character recognition since the first report in 1985. The pioneer work in on-line handprinted Thai recognition is described in [114] where structural analysis of character strokes, such as sequence, type and location of arcs together with convexity and concavity of contours, are extracted and used to classify the characters. The template matching process is used in the classification process. An on-line handwriting recognition system that can recognise Thai, English, numeral and symbol characters is presented in [115, 116]. In that system, a vector sequence that represents the sequence of directions between each successive sample in a character stroke is used as the feature. It is then passed to three sets of neural networks as an input. These three neural-networks units are responsible for the baseline and non-baseline Thai characters and the English characters, respectively.

On-line Thai handwriting recognition has received more interest over the past few years, and many different approaches are employed. In [117], the features extracted from each stroke segment of the on-line Thai characters are used as the inputs of the HMM-based recogniser. Similar features with the use of neural-networks classifiers, rather than HMMs, are presented in [118]. The work in [119] determines distinct features of the Thai characters which are combined to construct compound features. These features are useful only for the recognition of handprinted and well written characters. The research in Thai handwriting recognition is still in its early stages. Considerably more research is required to realise full benefit for the Thai people.

## 5.4 Summary

Most implementations of Thai character recognition rely on the presence of distinct characteristics of Thai characters such as loops and curls, which can be clearly seen in machine-printed or handprinted characters. However, these features are often omitted in an unconstrained writing style; loops are not fully closed or are written as blobs; curls are likewise elided or attenuated. These techniques are not suitable to recognise the unconstrained-style handwriting. This chapter introduces an overview of the characteristics of the Thai alphabet and its writing style, including a review of previous work in Thai character recognition. Research in Thai character recognition has progressed slowly compared to other scripts. This is partly due to its late start. This chapter is essential as a background to thoroughly understand our proposed techniques, based on discriminative training and HMMs, for Thai handwriting recognition in Chapter 6 which aims to solve the aforementioned problems and investigate various aspects of discriminative training.

# Chapter 6

# MMI Training of HMM-Based Thai Handwriting Recognition

## 6.1 Introduction

As discussed in Chapter 5, the Thai characters pose a unique challenge for automatic handwriting recognition on the grounds that there are several groups of Thai characters which have similar shapes, and vary only in small details such as loops and curls. The majority of previous studies rely on the presence of these details as important clues to indicate the identity of the character. However, in an unconstrained-writing style, loops are not fully closed or are written as blobs; curls are likewise elided or attenuated. Our aim is to solve this problem by not relying on the precise formation of loops and curls, and to make use of efficiently computed global features with a trainable classifier, instead of searching for specially defined local structural features. In this chapter, we demonstrate the use of discriminative training and propose feature extraction techniques to improve the performance of the HMM-based Thai handwriting recognition system. The performance of these techniques is also compared with that of an high-performance classifier, the convolutional networks. Although HMMs are generally used for cursive handwriting recognition, their ability to capture statistical variations of natural handwriting is useful for isolated character recognition. In addition, it is anticipated that the discriminative training techniques reported in this thesis would be applicable to a more difficult task of cursive handwriting recognition (see Section 9.2.1).

Because automatic Thai handwriting recognition is still in the early stages of research, most researchers are required to collect their own data. Section 6.2 describes the database of handwritten Thai characters used in this research and its collection procedure. Prior to applying discriminative training to the HMMs, the feature extraction techniques that perform reasonably well on the ML-trained system need to be put in place. An investigation of various feature extraction techniques and their performance is discussed in Section 6.3. These include an introduction to the novel techniques based on block-based PCA and composite images, and their evaluations on different HMM structures. Extensive studies of MMI training in the HMMs are presented in Section 6.4. We focus on a variation of the Extended Baum-Welch algorithm introduced by Povey [10]. Section 6.5 summarises the results of MMI training and investigates other useful techniques to further improve its performance. The discussion and summary of the chapter are addressed in Section 6.6.

## 6.2 Database of Thai Handwriting

The standard Thai writing style is in block letters, and each character is not connected together, but separated by a small space in between (see Section 5.2). Nevertheless, the formation of touching characters may occur when writing is done quickly and carelessly. The issue of recognising touching characters requires a further segmentation process [98, 99, 100, 112] which is beyond the scope of this research. Thus, the database of isolated Thai characters collected from multiple subjects is adequate and appropriate.

Our main focus is off-line Thai character recognition. Ideally, our experiments would have been conducted using the standard database to provide results which can be compared with the results quoted in other systems. In 2003, the Thai government's science research organisation, NECTEC (National Electronics and Computer Technology Center), released the standard database for *on-line* Thai handwriting recognition research, which is regarded as the first of its kind.[1] This database was not available when this research began in 2000, and thus most researchers needed to collect the primary data themselves. As a result, we have collected Thai handwriting samples and subsequently constructed the database to be used in our experiments.

A form for collecting handwriting data was specifically designed. It contains several empty boxes for handwriting samples to be filled in. Each box corresponds to each Thai character. Out of 87 Thai characters, only 77 characters are in our focus because the other ten characters are obsolete or rarely used in common writing. The so-called *baseline* characters are composed of consonants, numerals and some vowels which are written on the baseline. There are 64 baseline characters in total. The remaining 13 characters are the *non-baseline* characters which are written above or below the baseline. These include tone marks placed above and below the vowels.

Copies of this form were distributed to 20 native Thai writers, who in 2001 were studying at universities in England, France and Germany. The subjects were allowed to write in their own style and contributed at least five to nine handwriting samples for each character. The forms were then scanned on a high-quality flatbed scanner at 300 dots per inch resolution and stored as 8-bit greyscale images. First, the pre-processing techniques, such as noise removing and thresholding, were applied to the scanned forms. Figure 6.1 shows a part of the form after the pre-processing stage. Next, the character images were extracted from the boxes using the standard two-pass algorithm [120] in which a label was assigned to each pixel in the first pass, with label equivalence based on the eight-neighbour connectivity. Equivalent classes were determined, and a second pass updated each pixel in a connected component with a label unique to that component. Strokes intersecting

[1]http://arnthai.nectec.or.th/resource.asp

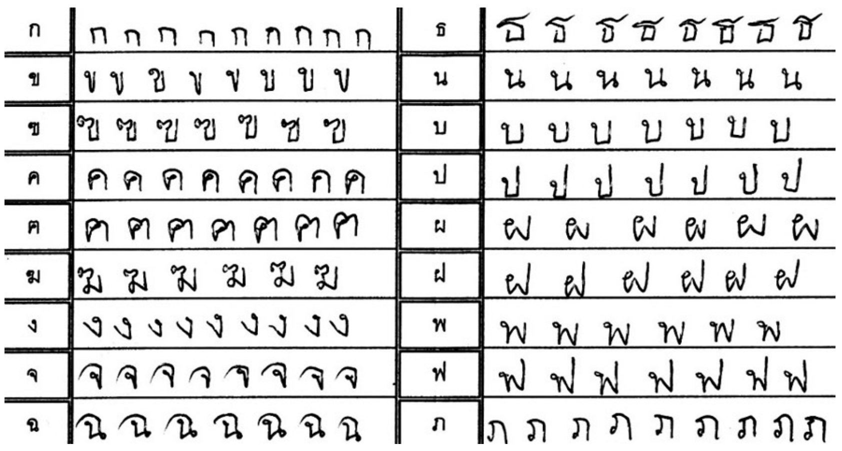

ก
ข
ฃ
ค
ฅ
ฆ
ง
จ
ฉ
ธ
น
บ
ป
ผ
ฝ
พ
ฟ
ภ

**Figure 6.1:** A part of the form for collecting handwriting samples from a native Thai writer.

| Character Categories | Number of Characters | Number of Samples | |
|---|---|---|---|
| | | Training Samples | Testing Samples |
| Baseline Characters | 64 | 3,840 | 4,195 |
| Non-Baseline Characters | 13 | 780 | 760 |
| Total | 77 | 4,620 | 4,955 |

**Table 6.1:** Details of the Thai handwriting database used in the experiments

pre-printed lines on the form were adjusted manually. Additional 5-pixel white space was applied to a bounding box of the characters. Finally, the aspect ratios and sizes of the resulting images were normalised to give bi-level images of 64×64 pixels. As a result, there were approximately 130 samples for each character. The database was divided into the training and testing sets with approximately equal size. The first three handwriting samples for each particular character from all writers were selected to be the training set, while leaving the rest to be the testing set. The number of samples in the database is summarised in Table 6.1.

## 6.3 Feature Extraction and Evaluations of the ML-trained System

Unlike *on-line* handwriting where temporal information (the trajectory of the pen input) is available, *off-line* handwriting is usually represented by scanned images which do not convey dynamic information. When the HMM-based recogniser is applied to off-line character recognition problems, it is necessary to convert the static character image into a sequence of observations as required by the HMMs. The conversion process can be carried out by two different approaches: *analytical* and *sliding-window* approaches. The analytical approach relies on topological features that describe structural properties of the characters. For example, Bunke *et al.* [61] and Khorsheed [121] extract a sequence of edges and loops from the skeleton graph of a word. In contrast, the method proposed by El-Yacoubi *et al.* [68, 122] employs sets of features in accordance with the shape of the character such as its contour, ascender, descender and loops. This approach is deemed inapplicable to our recognition tasks because many Thai characters have a relatively similar shape which can be distinguished by only small details. Moreover, some important details in the characters may be diminished during the skeletonisation process.

On the other hand, the sliding-window approach is a more general method which is widely used in the literature, both cursive [4, 62, 123, 124] and isolated character recognition [125]. In this instance, a narrow window is applied over the character image. The portion of the image covered by the window, called a *frame*, is captured while it is moving from left to right, sometimes from top to bottom or in diagonal directions [126]. A series of image frames is obtained as a result. An advantage of this approach is that it allows various types of features to be extracted from the image frame, not restricted to only the structural features. The left-to-right sliding direction is also appropriate for the HMMs with the left-to-right topology.

In this research, we utilise the sliding-window approach. The window which has the width of $w$ pixels and the height of character images, $H$, is moved from left to right with a step size of $o_h$ pixels. Given a character image of size $W \times H$, every two successive frame overlaps by $w - o_h$ pixels, and a sequence of $\frac{W-w}{o_h} + 1$ frames is generated. The optimal values of the window's width and the step size are task- and data-dependent, and thus can only be determined by experiments. The size of character images from the Thai handwriting database in Section 6.2 is $64 \times 64$ pixels ($W = H = 64$). In our preliminary experiments [127], setting $w$ and $o_h$ to 4 and 1, respectively, gives reasonable recognition results. As a consequence, the character image is transformed into a sequence of 61 frames (size $4 \times 64$ pixels), each of which is delivered to a subsequent feature extraction stage. Figure 6.2 shows a process of generating frames from the character image by using the sliding-window technique.

Since there is no guideline for feature extraction in the HMM-based Thai handwriting recognition system, the experiments described in the following sections are conducted to evaluate various feature extraction methods, and thus our proposed techniques to improve the recognition performance are presented accordingly.

### 6.3.1 Evaluations of Feature Extraction Methods

After operating the sliding-window technique, a feature extraction process is carried out to construct a feature vector from each of the image frames. The feature vector should effectively describe formations of pixels in the frame and, at the same time, be invariant to the small amount of noise. The feature extraction process aims to create as compact a representation as possible that can still be used to accurately classify the character. The evaluations of feature extraction centre on three different schemes, namely pixel, moment-based and the Gabor wavelets features, which can be summarised as follows:

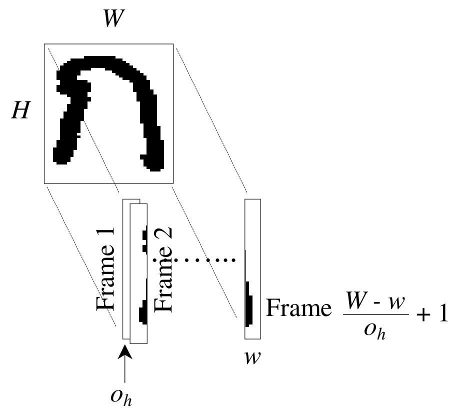


**Figure 6.2:** The sliding-window technique used to generate a sequence of frames from the character image

### Pixel Features

The simplest way to extract features from the image frame is to use its raw pixel data. Therefore, a feature vector with 256 components is extracted from each of the 4 × 64-pixel frames. This results in a high dimensional feature vector which is very sensitive to small distortions of the input image. Pattern recognition in the space containing a large number of features is subject to the so-called *curse of dimensionality*, indicating that the quantity of training data used to specify the decision boundaries for the classification problem grows exponentially with the number of dimensions [12]. More compact features are discussed in the following sections.

### Moment-Based Features

Moments provide an efficient way of describing the image function, $f(x, y)$. They have been used for many different image analysis and object recognition tasks, including character recognition [128]. The *central* moments of order $p+q$ for an $w \times H$-pixel image are given by:

$$\mu_{pq} = \sum_{x=1}^{w}\sum_{y=1}^{H}(x-\bar{x})^p(y-\bar{y})^q\, f(x,y), \tag{6.1}$$

where $\bar{x} = \frac{\sum_x \sum_y xf(x,y)}{\sum_x \sum_y f(x,y)}$ and $\bar{y} = \frac{\sum_x \sum_y yf(x,y)}{\sum_x \sum_y f(x,y)}$. Low-order moments represent a gross image shape whereas the higher-order moments reflect details in the image. A simple feature vector is constructed from the central moments of order up to 3. Note from Eq. 6.1 that both $\mu_{10}$ and $\mu_{01}$ are zero, so they are excluded here. As a result, the image frame is transformed into the 8-dimensional feature vector, i.e. $\{\mu_{00}, \mu_{11}, \mu_{20}, ..., \mu_{30}, \mu_{03}\}$.

The earliest and most notable work on the moment features is done by Hu [129]. Hu demonstrated that various combinations of central moments could produce representations that were invariant to translation, scale and rotation. Here, seven moment invariant features [88] derived from nonlinear combinations of the central moments are generated from the frame. Despite their numerous successful applications, the major drawback of

Hu's invariant moments is that they cannot be formulated from higher-order geometric moments. That is to say, their components are based on the geometric moments which are non-orthogonal. These become highly correlated as the order increases. An alternative method is to use orthogonal moments, such as the $p + q$ Legendre moments [130]. The feature vectors constructed from these moments are investigated. In addition, we also evaluate the affine moment invariants (AMIs), as proposed by Flusser and Suk [131], which are invariant under general affine transformation.

### Gabor Wavelets Features

An application of 2-dimensional Gabor filters in computer vision was pioneered by Daugman [132, 133]. In his early work, the Gabor filters were proposed as a representation of receptive field properties of neurons in visual cortex, as well as being the operators for practical image analysis problems. A family of 2-D Gabor filters can be used as an expansion basis to represent images. Because such Gabor filters are parameterised to be dilates and translates of each other, i.e. deriving from a common template, they constitute a *wavelet* basis and, therefore, are known as 2-D Gabor wavelets. The Gabor wavelet representation describes the orientation structure in the image while preserving information about its spatial relations. The Gabor filters have been widely used for image analysis tasks, such as texture analysis and iris recognition. Recently, a variant of 2-D Gabor filters proposed for face recognition [134] has been successfully applied to a number of character recognition problems [44, 135]. In this instance, the 2-D Gabor filter is defined as:

$$G(x, y; \omega, \sigma, \theta_k) = \frac{\omega^2}{\sigma^2} \exp\left(-\frac{\omega^2(x^2+y^2)}{2\sigma^2}\right) \left[\exp\left(i\,\omega R\right) - \exp\left(\frac{\sigma^2}{2}\right)\right], \tag{6.2}$$

where

$$R = x\cos\theta_k + y\sin\theta_k. \tag{6.3}$$

This describes a complex sinusoidal of particular frequency and orientation, modulated by a 2-D Gaussian shaped function, known as an *envelope.* The parameter $\sigma$ defines the standard variation of the Gaussian envelopes. A property of the sinusoidal wave is specified by $\omega = \frac{2\pi}{\lambda}$ and $\theta_k$, where $\lambda$ and $\theta_k$ serve as its wavelength and orientation, respectively. By convolving the Gabor filter with an $w \times H$ frame image at a particular sampling point $(x_0, y_0)$, the magnitude of the response is regarded as the Gabor feature, $g_{\sigma,\omega,k}(x_0, y_0)$, i.e.

$$g_{\sigma,\omega,k}(x_0, y_0) = \left| \sum_{x=-x_0}^{w-x_0-1} \sum_{y=-y_0}^{H-y_0-1} f(x+x_0, y+y_0)\, G(x, y; \omega, \sigma, \theta_k) \right|. \tag{6.4}$$

The value of $\theta_k$ is given by $\theta_k = \frac{k\pi}{M}$ where $k = 0, 1, 2, ..., M-1$, and $M$ denotes the number of orientations. Therefore, a total of $M$ Gabor features can be derived from one sampling point, each of which corresponds to the filter with orientation $\theta_k$. These comprise the orientation-dependent and localised spatial information of the image. Having uniformly chosen $N_x \times N_y$ sampling points from the $w \times H$ frame image, a total of $N_x \times N_y \times M$ Gabor features can be extracted from a given frame to create a feature vector accordingly.

We fix $\sigma = \pi$ and $\lambda = 8$, which appear to yield good results. The optimal number of sampling points and orientations needs to be obtained experimentally. Because the frame has the very narrow width compared to its height, the distribution of pixels and their shapes along the vertical direction provides an important clue for character discrimination. Therefore, we choose to sample $N_y$ points only along the vertical dimension ($N_x = 1$). We evaluate various settings of $N_y$ and $M$ between 2 and 8, which are denoted by $Gabor(N_y,M)$ in the following experiments.

**Recognition Performance**

The training and evaluation of the HMMs are carried out by using the HTK toolkit (a toolkit primarily used for speech recognition [136]) with additional implementations to prepare the feature vector sequences from the character images and to train models using discriminative training (see Sections 6.4 and 6.5). The HMMs employ a left-to-right topology, in which each state has a transition to itself and the next state, as shown in Figure 3.1. HMM states have diagonal Gaussian probability density functions (pdfs). At first, the experiments are conducted on the ML-trained system to investigate feature extraction methods, and to provide reliable initial models for discriminative training. A simple structure of the HMMs is utilised, i.e. one Gaussian mixture per state with no parameter tying. The recognition setup is based on the notion that one would have prior information (from the position of the characters) of which characters are baseline characters and which ones appear above or below the baseline characters, i.e. non-baseline characters. Therefore, the recognition of both character groups is done separately.

The classification experiments were performed using several different feature vectors assembled from the aforementioned features. The character sample is converted into a sequence of 61 feature vectors as an input of the HMMs. Different feature extraction methods yield feature vectors with different dimensionalities. In the experiments, the number of HMM states is varied to find the optimal model configuration. The optimisation of the HMMs is done by ML training using the Baum-Welch algorithm. Training is executed for 40 iterations. The average recognition accuracies of both training and testing sets are measured. The recognition results of the testing set for various feature extraction techniques are presented in Table 6.2.

An increase in recognition rates is obtained with respect to an increase in the number of HMM states when the HMMs have less than 40 states. However, only a slight improvement (sometimes a small decrease) in recognition accuracies is achieved when the number of states exceeds 40. This indicates that the character model with 40 states tentatively provides optimal recognition performance and serves as a good representation of our character sample, which appears as a sequence of 61 frames. The results also show that a simple feature such as the pixel feature is effective by yielding up to 85.55% accuracy, despite its high dimensionality. These are higher than the best result of the moment-based features, which yields only 84.94% accuracy. The feature extracted from a (narrow) frame should be able to describe the distribution of pixels along the vertical direction. Because most of the moment features we use are invariant to the translation of pixels, the frames that contain similar shapes, albeit locating at different places, may

| Feature Extraction Methods | Dimen-sions | Recognition Results (%) | | | | |
|---|---|---|---|---|---|---|
| | | 6 states | 12 states | 25 states | 40 states | 50 states |
| Pixel Features | 256 | 53.64 | 74.97 | 83.96 | 85.55 | 85.39 |
| Central Moments | 8 | 28.90 | 66.42 | 76.87 | 77.56 | 77.95 |
| Hu's Moment Invariants | 7 | 20.12 | 51.64 | 62.54 | 65.93 | 64.74 |
| Legendre Moments | 10 | 37.48 | 78.91 | 84.56 | 84.94 | 82.87 |
| Affine Moment Invariants | 4 | 11.75 | 18.02 | 38.30 | 42.42 | 41.15 |
| Gabor(4,2) | 8 | 55.64 | 75.80 | 82.89 | 85.25 | 84.88 |
| Gabor(4,4) | 16 | 61.92 | 77.60 | 84.26 | 85.41 | 85.51 |
| Gabor(4,8) | 32 | 61.29 | 76.61 | 83.47 | 84.50 | 85.09 |
| Gabor(8,2) | 16 | 62.56 | 79.74 | 85.89 | 87.79 | 87.41 |
| Gabor(8,4) | 32 | 67.25 | 80.42 | 86.74 | 88.42 | 87.81 |
| Gabor(8,8) | 64 | 67.23 | 80.36 | 86.34 | 87.91 | 87.31 |
| Gabor(16,2) | 32 | 56.91 | 76.95 | 83.53 | 85.37 | 85.03 |
| Gabor(16,4) | 64 | 60.89 | 78.18 | 83.92 | 85.95 | 86.12 |
| Gabor(16,8) | 128 | 61.05 | 77.64 | 83.55 | 85.31 | 85.49 |

**Table 6.2:** Recognition results of the testing set for different feature extraction methods and various number of HMM states

produce an identical feature vector. Therefore, the moment invariant features may not be appropriate for discriminating characters through the use of the sliding-window approach.

Among three feature extraction methods, the results show that the Gabor features outperform the others, with the best recognition rate of 88.42% on unseen samples. This suggests that the Gabor features are capable of capturing more of the underlying structures in the image while preserving their spatial relations, which is a desirable property for a feature extraction technique in the sliding-window approach. With a handful of dimensions (for example, 16 dimensions from the *Gabor*(4,4) feature), we can achieve the results equivalent to that using the 256-dimensional pixel features. However, the performance of the Gabor features varies in accordance with the number of sampling points and orientations. A comparison of pixel features and various settings of Gabor features for the 40-state HMMs is illustrated in Figure 6.3. Having stated that the Gabor features yield the best results, the experiments reported in the following sections are based on these features.

### 6.3.2 PCA and Block-Based PCA

Principal Component Analysis (PCA) [137] is a well known statistical technique, commonly used in pattern recognition, for dimensionality reduction and data analysis. It is also successfully applied to speech and character recognition applications [138]. The basic

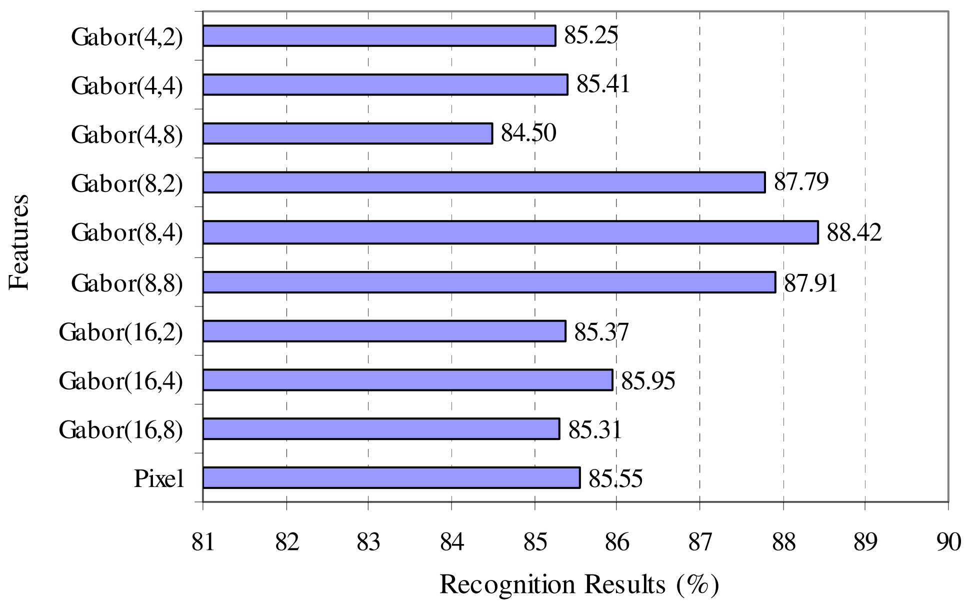


**Figure 6.3:** Recognition results of the testing set for pixel features and various settings of Gabor features in the 40-state HMMs, summarised from Table 6.2

principle of PCA is to find a linear transformation matrix $\mathbf{A}$ to convert a set of *correlated* data into an *uncorrelated* one. The data are considered uncorrelated when the off-diagonal components of their covariance matrix are zero. Unlike Independent Component Analysis (ICA) or Linear Discriminant Analysis (LDA) [137] which attempts to discriminate the data of each category, PCA does not take into account category labels and data classification during the transformation process. Given the $d$-dimensional feature vector $\mathbf{x}$, the transformation of $\mathbf{x}$ into a new feature space $\mathbf{y}$ by PCA can be written as:

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}), \tag{6.5}$$

where $\boldsymbol{\mu}_{\mathbf{x}}$ represents the mean vector derived from the entire feature vectors of the training set. The covariance matrix $\mathbf{C}_{\mathbf{x}}$ of the same data is also calculated. The transformation matrix $\mathbf{A}$ is created by ordering the *eigenvectors* of $\mathbf{C}_{\mathbf{x}}$ in a way that their *eigenvalues* are descending. Because the first few eigenvectors of $\mathbf{C}_{\mathbf{x}}$ (with the highest eigenvalues) indicate the directions where the training data have large variances, it is possible to discard the components whose eigenvalues are small without losing a significant amount of information. The simplest application of PCA is concerned with projecting the first $d'$ principal components of the $d$-dimensional feature vector into a $d'$-dimensional vector, which is referred to as the standard PCA technique.

As an alternative to PCA, we propose a technique called *block-based PCA* which has demonstrated a slight improvement of recognition accuracies compared with the standard PCA [139]. Its design stems from the earlier discussion that the distribution of pixels and their shapes along the vertical direction provides an important clue for character discrimination. Thus, the finer features we extract from the vertical direction, the better discrimination these features possibly become.

In block-based PCA, we begin with the $w \times H$-pixel frame image. This is divided vertically into overlapping *blocks* of the height $h$, with successive blocks moving by a

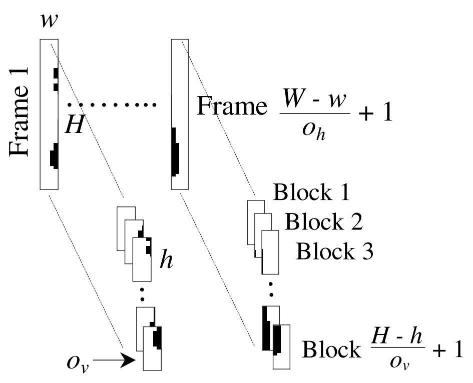


**Figure 6.4:** The process for dividing each image *frame* into several *blocks*

vertical offset $o_v$ until they cover the entire vertical height (see Figure 6.4). As a result, a total of $\frac{H-h}{o_v} + 1$ blocks are obtained from each frame. Given the 4 × 64 image frame, we specify the block height to 16, and each block overlaps by an 8-pixel offset. Therefore, it requires 7 blocks to cover a frame of height 64. The designated features are then extracted from each of the 16 × 8-pixel blocks, resulting in a $d$-dimensional feature vector. The vectors from all 7 blocks are concatenated to form a large feature vector of size $7d$. Finally, PCA is applied to reduce its dimension to a $d'$-dimensional vector.

Another way to perform block-based PCA is to use an individual transformation matrix for each block to extract its principal components, which is calculated from the covariance matrix of that block in the training data. Each block is projected by PCA to $\bar{d}$ dimensions. These are concatenated for all 7 blocks to make a final vector of size $d' = 7\bar{d}$. Note that, in this case, the final dimension has to be multiplicities of block numbers by which it is not as convenient as the aforementioned method. Moreover, our preliminary experiments reveal that the performance of both techniques do not vary significantly. Therefore, the results shown in the following sections are based on the former technique.

A series of experiments to evaluate the performance of standard PCA and block-based PCA is conducted. Since Gabor feature vectors are relatively small, we do not alter their dimensions during the transformation process, i.e. $d' = d$. However, the dimensions of pixel features are reduced from the original 256 to 42 ($d' = 42$). Note that the transformation matrices are computed by using only the feature vectors from the training data. Once all feature vectors are transformed, they are subsequently used in the training and recognition processes of HMMs. The settings of training parameters are similar to those of Section 6.3.1, but only the HMMs with 40 states are evaluated. The recognition performance is summarised in Table 6.3 and compared with the results in Figure 6.3 (without PCA). These results show that both the standard PCA and block-based PCA techniques are efficient and can increase recognition accuracies of the system. Our proposed block-based PCA slightly outperforms the standard PCA technique in all cases. Further investigations and results of PCA and block-based PCA are presented in Section 6.3.3.

| Feature Extraction Methods | Without PCA | | Standard PCA | | Block-Based PCA | |
|---|---|---|---|---|---|---|
| | Dim | Recognition Results | Dim | Recognition Results | Dim | Recognition Results |
| Gabor(4,2) | 8 | 85.25 % | 8 | 86.01 % | 8 | 87.81 % |
| Gabor(4,4) | 16 | 85.41 % | 16 | 87.18 % | 16 | 88.11 % |
| Gabor(8,2) | 16 | 87.79 % | 16 | 88.27 % | 16 | 88.49 % |
| Gabor(8,4) | 32 | 88.42 % | 32 | 88.98 % | 32 | 89.10 % |
| Pixel Features | 256 | 85.55 % | 42 | 85.89 % | 42 | 86.42 % |

**Table 6.3:** Recognition results of the testing set when using different PCA techniques

### 6.3.3 Composite Images

In addition to extracting features from the normalised character images, we also evaluate our feature extraction techniques on the transformed versions of those images, such as the polar transformation and 90-degree rotation. The polar transformation (similar to the log-polar transform [140]) is widely used in computer vision research and, recently, in character recognition [121]. It is a conformal mapping from points in the image $f(x, y)$ to points in the polar image $g(r, \theta)$. By defining an *origin* $O = (o_x, o_y)$ which is given by the centroid $(o_x = \bar{x}, o_y = \bar{y})$ of the image, the mapping is described by:

$$r = \frac{\sqrt{(x - o_x)^2 + (y - o_y)^2}}{d}, \tag{6.6}$$

$$\theta = \arctan\left(\frac{y - o_y}{x - o_x}\right), \tag{6.7}$$

where $d$ is defined as the maximum distance between $O$ and all pixels in $f$. The maps are then normalised to the size of 64 × 64 pixels. Examples of the original image, together with its polar transform and clockwise 90-degree rotated versions are illustrated in Figures 6.5 (a), (b) and (c), respectively. Note that these transformations are operated on the entire image, not at the frame level.

The sliding-window and feature extraction techniques are applied to these images in the same fashion as described in Section 6.3.1. Our preliminary experiments [127] show that the original image and the two transformed images give comparable recognition results individually. However, their confusion matrices exhibit significant differences between each of the character classes, meaning that each transformation is capable of capturing different important information of the characters. These bring in an idea of using a *composite image* in which the original image is concatenated with the 90-degree rotation of itself and the polar transformed version of itself, as depicted in Figure 6.5 (d). All character samples are converted into the composite images which are later used in the training and recognition processes.

We evaluate the performance of composite images on two selected features: *Gabor*(4,4) and *Gabor*(8,4). Since the number of frames obtained from the composite images is three times larger than that of the original images, the appropriate number of

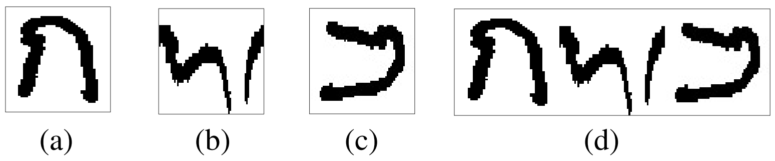


**Figure 6.5:** Samples of transformed character images: (a) original image, (b) polar transformed image, (c) clockwise 90-degree rotated image and (d) composite image

| Feature Extraction Methods | Dim | Original Results (%) | Recognition Results (%) when using Composite Images | | | | |
|---|---|---|---|---|---|---|---|
| | | | PCA Methods | 40 states | 50 states | 60 states | 70 states |
| Gabor(4,4) | 16 | 85.41 | Without PCA | 88.98 | 89.67 | 89.97 | 90.49 |
| | | | Standard PCA | 89.38 | 90.19 | 90.54 | 91.02 |
| | | | Block-Based PCA | 89.73 | 90.33 | 90.70 | 91.10 |
| Gabor(8,4) | 32 | 88.42 | Without PCA | 89.67 | 90.70 | 91.16 | 91.71 |
| | | | Standard PCA | 90.42 | 91.22 | 91.43 | 91.75 |
| | | | Block-Based PCA | 90.53 | 91.40 | 91.65 | 92.03 |

**Table 6.4:** Recognition results of the testing set for different PCA techniques and various number of HMM states when composite images are in used

HMM states needs to be sought. In addition, the applications of PCA and block-based PCA to the composite images are investigated. Their recognition results are summarised in Table 6.4, compared with the results of using normal images. The HMMs with 70 states seem to be the optimal topology for the composite images. The number of states exceeding 70 does not give significantly different performance, thereby being excluded in the table. From the results, the composite images are superior to the normal images in all cases. By using the composite images, we obtain a large improvement up to 35% relative (from the absolute 85.41% to 90.49%) depending on the experimental conditions. Both PCA techniques are also effective when applied to the composite images, and can increase recognition accuracies of the systems. At the same time, it is shown that the block-based PCA is slightly better than the standard PCA technique, in support of the results obtained from Section 6.3.2.

Thus far, we have studied the performance of feature extraction in ML training by using a simple HMM structure with a single Gaussian per state and without parameter tying. An additional set of experiments is conducted to evaluate the performance of mixture-Gaussian systems, both with and without parameter tying. We achieve a small increase in the recognition accuracies when multiple Gaussians per state are utilised. These improvements come with a trade-off for an increasing number of system parameters, which however can be solved by tying the Gaussian mixtures. Nevertheless, such systems require much more computation time in the recognition process than the simple structure HMMs, due to the number of Gaussians involved in likelihood computations. Therefore, the single state HMMs are deemed more appropriate to be considered as the initial models for MMI training. Moreover, the results from Chapter 4 show that MMI training is more effective when applied to the untied systems. For this reason, the experiments of MMI training reported in Section 6.5 are based on the simple structure of HMMs that uses the composite images and block-based PCA described in this section.

## 6.4 MMI Training in HMMs

Three different optimisation algorithms of MMI training in the HMMs have been evaluated (see Section 4.4). Although the QuickProp algorithm is selected for MMI training due to its highest convergence rate, it poses some difficulties in specifying the learning parameters when the multi-state HMMs are put in place. As the number of model parameters grows, the algorithm becomes more sensitive towards the settings of the learning rate, $\eta$, and $\varphi$. The optimal values of these variables for each trained model parameter need to be acquired. On the other hand, the Extended Baum-Welch algorithm demonstrates more stability and less sensitivity to the settings of learning parameters than the QuickProp algorithm, despite its slow convergence rate. It requires only a single learning parameter setting which is the smoothing factor, $D$. The convergence-guaranteed condition of the EBW algorithm is obtained when an infinite value of the smoothing parameter is used [71]. But, this results in a very small change of parameters. Recently, a variation of the EBW algorithm has been proposed by Povey [10]. Povey introduces a concept of *strong-sense* and *weak-sense* auxiliary functions which provide an efficient way to derive the EBW update rules for Gaussian parameters that are valid for any values of smoothing parameters. His approach also leads to the novel update rules for mixture coefficients and state transition probabilities. The algorithm is proven effective in large vocabulary speech recognition applications [10, 34]. The work of MMI training presented here focuses on this algorithm as it can efficiently determine the smoothing parameters. This method also yields fast convergence rates and requires only a small number of training iterations. Details of the algorithm are briefly described in the following sections.

### 6.4.1 Strong-Sense and Weak-Sense Auxiliary Functions

Parameter estimation in the HMMs involves either maximising or minimising the objective functions. However, the direct optimisation of these functions is complicated, and often it relies on an alternative method that employs *auxiliary functions*. For example, as discussed in Section 3.3.1, the use of the auxiliary function in ML training can simplify the problem of maximising the ML objective function. This auxiliary function is useful in such a way that it is an easy function to maximise rather than the actual ML objective function. An increase in its value is also bound to increase the ML objective function. Given the objective function $\mathcal{F}(\lambda)$ to be maximised, such an auxiliary function $\mathcal{Q}(\lambda, \lambda^{\tau})$ has the following property:

$$\mathcal{Q}(\lambda^{\tau+1}, \lambda^{\tau}) - \mathcal{Q}(\lambda^{\tau}, \lambda^{\tau}) \leq \mathcal{F}(\lambda^{\tau+1}) - \mathcal{F}(\lambda^{\tau}), \tag{6.8}$$

where $\lambda$ represents the model parameters that need to be estimated, $\lambda^{\tau}$ is the current model parameters treated as a constant, and $\lambda^{\tau+1}$ denotes the particular estimate of $\lambda$.

Starting with the current value of model parameters, $\lambda^{\tau}$, if the estimate $\lambda^{\tau+1}$ increases $\mathcal{Q}$, it will also increase $\mathcal{F}$. By iteratively maximising the auxiliary function $\mathcal{Q}$, it is guaranteed to reach a local maximum of $\mathcal{F}(\lambda)$. The auxiliary function that satisfies the condition in Eq. 6.8 is known as the *strong-sense* auxiliary function [10]. This kind of functions is used to maximise the ML objective function in the Baum-Welch algorithm. Nevertheless, it is not applicable to the maximisation of the MMI objective function in Eq. 3.24 which contains a negative sign of the denominator likelihoods because the inequality in Eq. 6.8 will no longer hold. Povey [10] introduces the concept of *weak-sense* auxiliary functions which have the following property:

$$\left.\frac{\partial\,\mathcal{Q}(\lambda^{\tau+1},\lambda^{\tau})}{\partial\lambda^{\tau+1}}\right|_{\lambda^{\tau+1}=\lambda^{\tau}} = \left.\frac{\partial\,\mathcal{F}(\lambda^{\tau+1})}{\partial\lambda^{\tau+1}}\right|_{\lambda^{\tau+1}=\lambda^{\tau}}. \tag{6.9}$$

The above equation is regarded as a minimum requirement for any auxiliary function to be used for optimising $\mathcal{F}$. Maximising the weak-sense auxiliary functions does not now guarantee an increase in $\mathcal{F}$. However, if there is no change in $\lambda^{\tau+1}$ after a number of training iterations, we can imply that a local maximum of $\mathcal{F}(\lambda^{\tau})$ has been reached, and the final $\lambda_{\tau+1}$ becomes the estimate of the model parameters. The weak-sense auxiliary function has to be carefully chosen to ensure good convergence.

## 6.4.2 EBW Re-Estimation Formulae

Povey derives particular weak-sense auxiliary functions suitable for the optimisation of the MMI objective function [10]. His approach yields the updating formulae for Gaussian parameters equivalent to the ones proposed in [71] (see Section 4.4.1). Analogous to Eqs. 4.12 and 4.13, the EBW formulae for Gaussian parameters updating in the CDHMMs can be written as:

$$\mu_{i,j,k,d}^{\tau+1} = \frac{(\Theta_{i,j,k,d} - \overline{\Theta}_{i,j,k,d}) + D_{i,j,k}\,\mu_{i,j,k,d}^{\tau}}{(\gamma_{i,j,k} - \overline{\gamma}_{i,j,k}) + D_{i,j,k}}, \tag{6.10}$$

and

$$(\sigma_{i,j,k,d}^{\tau+1})^2 = \frac{(\Psi_{i,j,k,d} - \overline{\Psi}_{i,j,k,d}) + D_{i,j,k}\left((\sigma_{i,j,k,d}^{\tau})^2 + (\mu_{i,j,k,d}^{\tau})^2\right)}{(\gamma_{i,j,k} - \overline{\gamma}_{i,j,k}) + D_{i,j,k}} - (\mu_{i,j,k,d}^{\tau+1})^2. \tag{6.11}$$

With the help of the weak-sense auxiliary functions, the EBW algorithm can be proven to converge under any values of the smoothing parameter $D$. This allows us to specify the smoothing parameter at the Gaussian level, hence $D_{i,j,k}$. In the implementation, the value of $D_{i,j,k}$ is set to the larger of i) twice the smallest value required to ensure that all variances are positive, or ii) $\overline{\gamma}_{i,j,k}$ times a further constant $E$ [10]. This provides a sensible approach to learning parameter settings, and is experimentally found to lead to good convergence. The value of $E$ is typically set to 1 or 2.

By applying the weak-sense auxiliary functions, he also derives a new EBW re-estimation formula for the parameters subject to the sum-to-one constraint (the mixture coefficients and state transition probabilities). Rather than computing the updated mixture coefficients directly from Eq. 4.4, another batch of iterative computations is conducted to estimate these updates after each MMI training iteration. For all models $\lambda_i$, state $s_j$ and mixture $k$, the value of $c^{(0)}_{i,j,k}$ is set to $c^{\tau}_{i,j,k}$ (the original value before a particular MMI iteration), and we iteratively compute $c^{(p)}_{i,j,k}$ by:

$$c^{(p+1)}_{i,j,k} = \frac{\gamma_{i,j,k} + D_{i,j,k}\, c^{(p)}_{i,j,k}}{\sum_m \gamma_{i,j,k} + D_{i,j,k}\, c^{(p)}_{i,j,k}}, \tag{6.12}$$

where

$$D_{i,j,k} = \left(\max_k \frac{\overline{\gamma}_{i,j,k}}{c^{\tau}_{i,j,k}}\right) - \frac{\overline{\gamma}_{i,j,k}}{c^{\tau}_{i,j,k}}. \tag{6.13}$$

During these sub-iterations, the occupation counts $\gamma_{i,j,k}$ and anti-occupation counts $\overline{\gamma}_{i,j,k}$ are not further collected. Therefore, their original values remain in use. It may require up to 100 sub-iterations ($p$) for the value of $c^{(p)}_{i,j,k}$ to converge. The final values of $c^{(p)}_{i,j,k}$ serve as the updated values of the mixture coefficients, i.e. $c^{\tau+1}_{i,j,k}$, for a particular MMI training iteration. A similar method can be applied to each row of the transition matrices.

### 6.4.3 Probability Scaling

From the MMI training experiments, we notice that the denominator likelihoods (the negative term in the MMI objective function) tend to be dominated by one of the character classes, which is usually the correct character. As a result, the other character classes become less likely to compete with the correct one. This may result in a classifier with non-smooth decision boundaries which are not generalised to the unseen samples. Schluter [73] introduces the smoothing parameter which serves as a scale on the log likelihoods to enable more characters to compete with the most likely character. The method has demonstrated some improvements on the recognition results of the testing set, as presented in [10, 34]. Consider the MMI objective function in Eq. 3.24, the probability scaling can be performed by:

$$f_{MMI}(\Lambda) = \frac{1}{R}\sum_{r=1}^{R}\left(\log\, P_\Lambda(\mathbf{O}^T_r|\mathcal{C}^r)^\kappa - \log\sum_{i=1}^{M} P_\Lambda(\mathbf{O}^T_r|C_i)^\kappa P(C_i)^\kappa\right), \tag{6.14}$$

where the smoothing parameter $\kappa$ is typically less than one.

To obtain an intuitive idea of probability scaling, consider a function $y = x^\kappa$ for various settings of $\kappa$ in Figure 6.6, where both $x$ and $y$ are bounded between 0 and 1. The smaller the value of $\kappa$ gets, the more significant the value of $x$ becomes. In this way, the MMI criterion increasingly takes into account the likelihood of the less likely characters. With $\kappa = 1.0$, the system functions as normal MMI training without involving a scaling factor. Although Povey [10] suggests that $\kappa$ should be within the range of $\frac{1}{10}$ to $\frac{1}{20}$ if no statistical language model is used, we find that the optimal value of $\kappa$ is highly dependent on the recognition tasks and needs to be obtained from the experiments. Note that probability scaling is applied to the forward-backward algorithm which calculates the occupation counts, $\gamma_{i,j,k}$ and $\overline{\gamma}_{i,j,k}$, for the HMM states [10]. This has no impact on the EBW algorithm, so the optimisation formulae in Section 6.4.2 can be used directly.

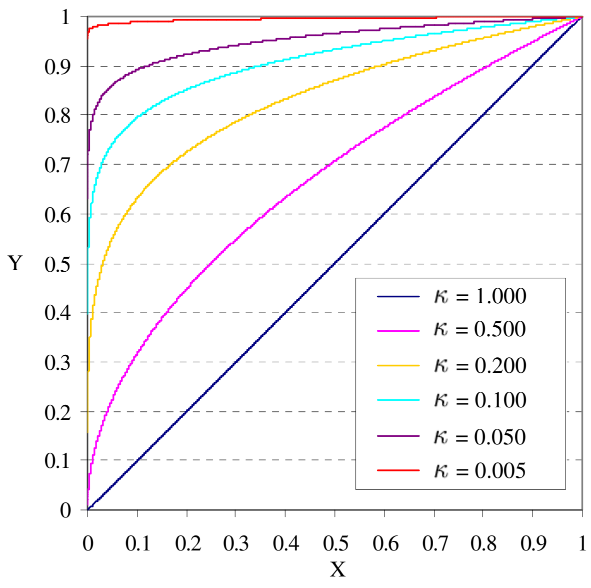


**Figure 6.6:** The graph showing a function $y = x^{\kappa}$ for different settings of $\kappa$

## 6.5 Experiments of MMI Training in HMMs

### 6.5.1 Performance of MMI Training

After the models are initialised by ML training, the MMI training experiments are subsequently conducted by using the EBW algorithm described in Section 6.4.2. The MMI training module developed for the HTK toolkit has been used in the work of Povey [10] and Nopsuwanchai and Povey [139]. In continuous speech recognition applications, the numerator and denominator likelihoods are efficiently computed via the use of *lattices*. The lattice stores alternate word sequences in the form of a graph where the arcs correspond to words, and the word sequences are encoded by the paths through the graph. A detailed description regarding lattices can be found in [10, 33]. However, in the case of isolated character recognition, the lattice contains only alternate character classes, but not their sequence information. The numerator lattice of each character sample represents its correct category while the denominator lattice encodes all possible classes of that sample. The denominator likelihoods of the baseline character samples are computed from all possible 64 baseline character classes, and in the same vein the possible 13 classes are used for the non-baseline characters.

The MMI training experiments are performed on the system that employs the *Gabor*(8,4) features with the composite images and block-based PCA. Although the ML-trained systems with 60 and 70 states yield slightly higher recognition accuracies than the ones with 50 states, our preliminary experiments find that the results of MMI training on the 50-state system are generally better than the other two. This may be due to the reason that discriminative training is more sensitive to the amount of training data than is ML training, as presented in [10] and [139]. The system with a higher number of model parameters would require higher number of training samples. Therefore, the following MMI training experiments are based on the 50-state HMMs. The value of $E$ in all experiments is set to 2, which happens to be the optimal value for our tasks.

In the first experiment, we evaluate the performance of MMI training without the effect of probability scaling, so the value of $\kappa$ is set to 1.0. MMI training is carried out for 10 iterations. All HMM parameters, e.g. the state transition probabilities, the means and variances of the Gaussians, are optimised by MMI. Figure 6.7 illustrates the MMI objective function varied with iterations of training, demonstrating that the MMI optimisation is proceeding smoothly. The recognition performance of both the training and testing sets is shown in comparison with the ML training results (see Figure 6.8). The improvement from MMI compared with the ML baseline on the testing set is equivalent to 15% relative (from 91.40% to 92.67%). However, it is interesting to note that the training set results rise quickly to 100% within only 6 iterations of MMI training while the testing set shows a more modest improvement. Even though MMI training is clearly efficient and can improve the results from ML training, this indicates that the system is not well generalised to the unseen data. This problem is accordingly addressed in Section 6.5.2.

### 6.5.2 Effects of Probability Scaling

The experiments in this section aim to investigate the effects of probability scaling towards the generalisation of the system. The scaling parameter $\kappa$ described in Section 6.4.3 controls the degree of contribution among competing categories and enables more characters to compete with the most likely character. Starting from the ML-trained system with the configuration similar to that of Section 6.5.1, a series of experiments with different settings of $\kappa$ is carried out. The recognition performance of the testing set for various values of $\kappa$ is summarised in Table 6.5 and Figure 6.9. The results do not vary significantly for $\kappa$ within the range of 1.0 and 0.05. However, an improvement of the testing set results becomes more dramatic when the value of $\kappa$ is below 0.02, as shown in Figure 6.9. The range of $\kappa$ between 0.003 and 0.006 provides a relatively high recognition performance, with the peak at $\kappa = 0.004$. The highest recognition accuracy reaches 95.74%, or equivalent to 50% relative error reduction when compared with the ML result.

| Values of $\kappa$ | Recognition Results |
|---|---|
| 1.00 | 92.69 % |
| 0.80 | 92.69 % |
| 0.60 | 92.67 % |
| 0.40 | 92.63 % |
| 0.20 | 92.59 % |
| 0.10 | 92.65 % |
| 0.08 | 92.63 % |
| 0.05 | 92.69 % |

**Table 6.5:** Recognition results of the testing sets for some values of the scaling parameters $\kappa$

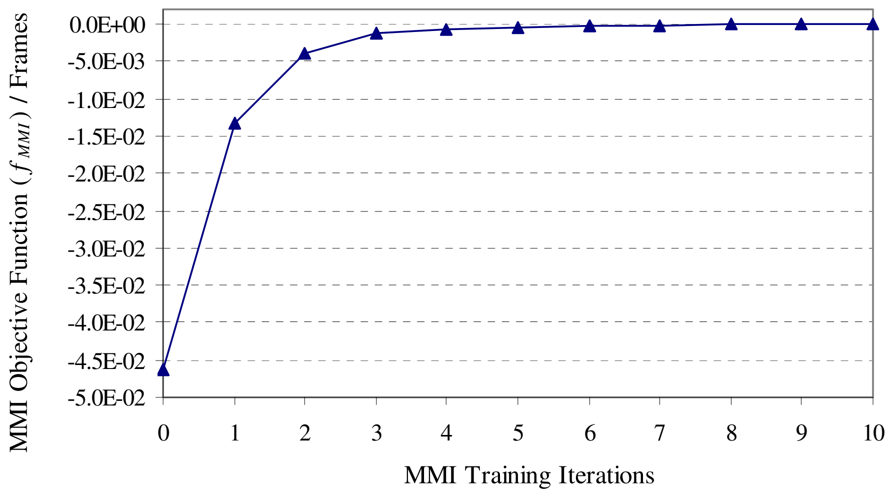


**Figure 6.7:** The growth of the MMI objective function versus the number of MMI training iterations

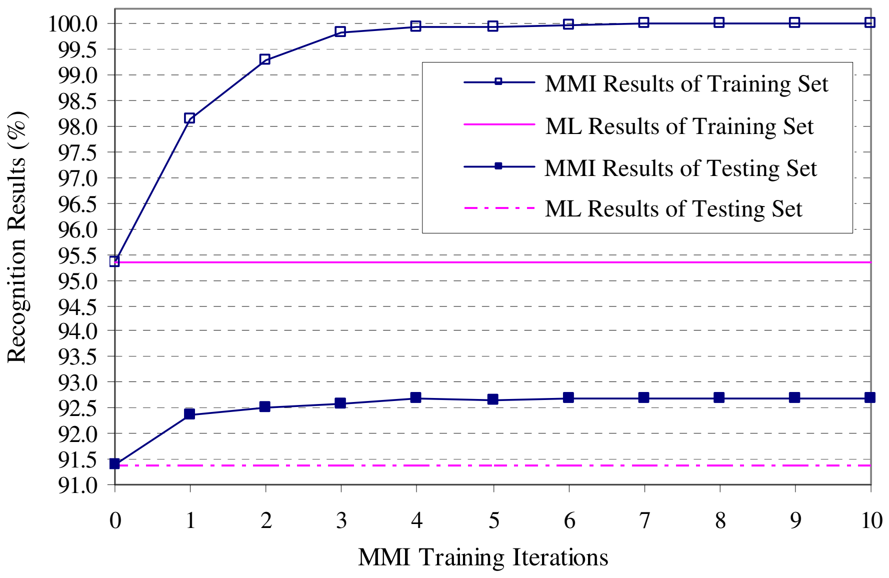


**Figure 6.8:** Comparison of the ML and MMI recognition results of both training and training sets

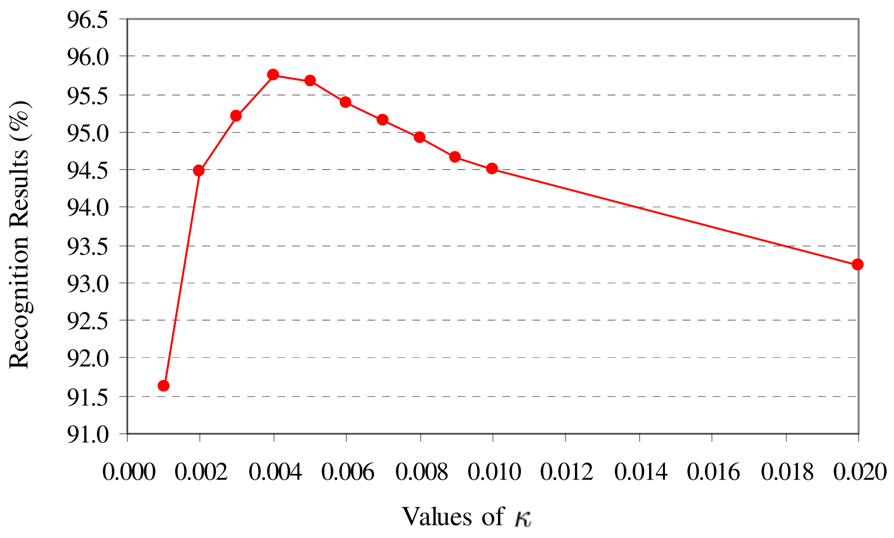


**Figure 6.9:** Recognition results of the testing set for different settings of the scaling parameters $\kappa$

Figure 6.10 shows the evolution of the MMI objective function versus training iterations for various settings of $\kappa$. The corresponding recognition accuracies of both training and testing sets are presented in Figure 6.11. With probability scaling, MMI training progresses less rapidly than when probability scaling is not in used, as seen from a gradual increase of the MMI objective function and the training set results. Nevertheless, this yields a swift improvement in the testing set performance. These results clearly demonstrate that probability scaling in MMI training is efficient and leads to good generalisation on the unseen samples. The optimal value of the scaling parameter is, however, highly dependent on the recognition tasks and the features used, and needs to be acquired from the experiments.

### 6.5.3 The Use of $N$-best Lists

Even though probability scaling helps more characters to compete with the most likely character, we find that the denominator likelihoods of only few character classes account for the sum of the denominator likelihoods in the MMI objective function. This finding allows us to compute the denominator likelihoods in a more efficient way. That is to say, instead of accumulating the probabilities from all possible classes, we can take into account only the $N$ most likely candidates specified by the $N$-best lists.

The recognition process generates the individual $N$-best list for each training sample. The denominator lattice of the sample is then constructed from the output of the $N$-best list. This process is repeated in each iteration of MMI training and throughout all training samples. In the experiments, we investigate the effects of various numbers of $N$ by using the same configuration as in the past two sections. The value of probability scaling $\kappa$ is set to 0.004. The recognition performance of the testing set for different values of $N$ is illustrated in Figure 6.12. It is shown that using the $N$-best lists to compute denominator likelihoods with $N \geq 3$ does not degrade recognition accuracies from the original result

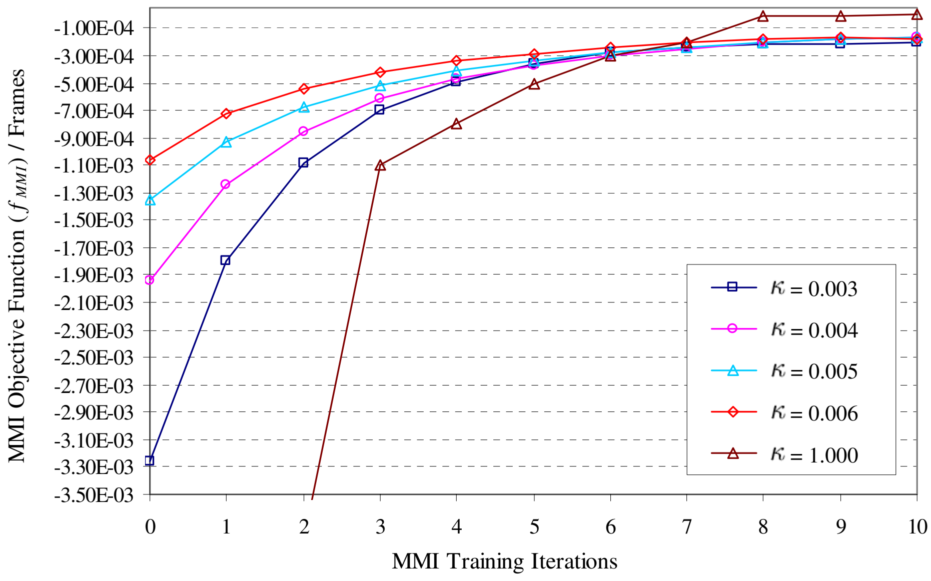


**Figure 6.10:** The growth of the MMI objective function versus the number of MMI training iterations for various settings of the scaling parameters $\kappa$

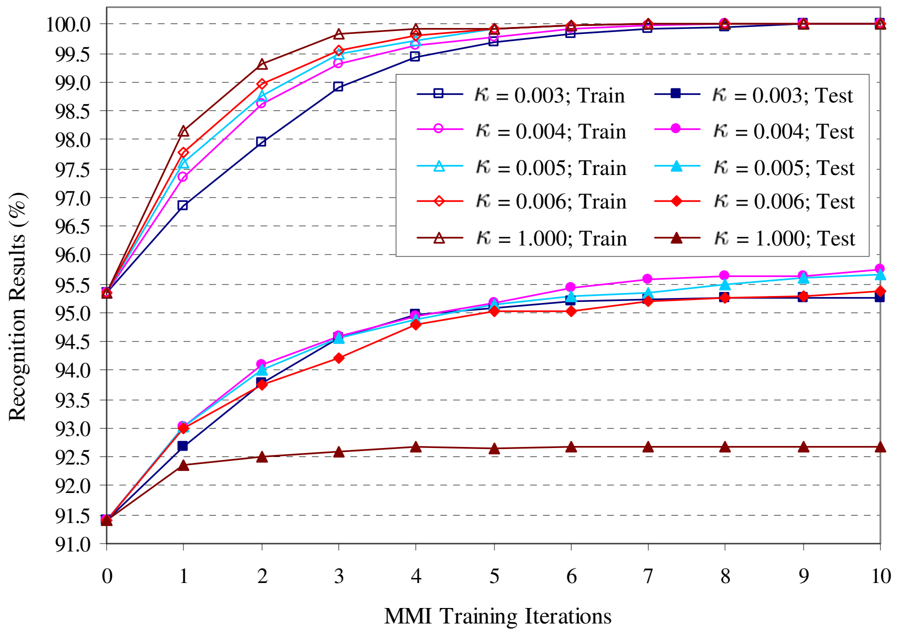


**Figure 6.11:** Recognition results of both training and testing sets versus the number of MMI training iterations for various settings of the scaling parameters $\kappa$

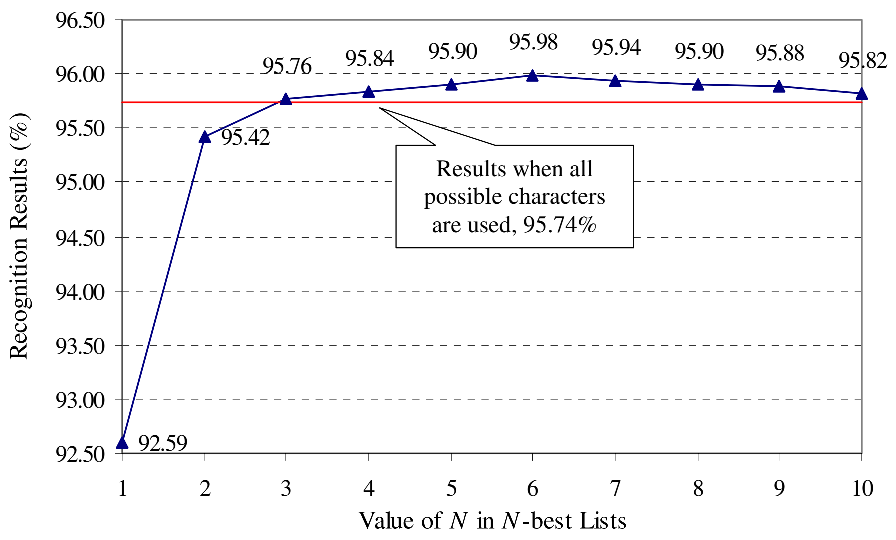


**Figure 6.12:** Recognition results of the testing set for different values of $N$ when the $N$-best lists are used, compared with the result of using all possible characters

of using all possible classes. It removes the confusing competing classes and hence can significantly reduce the computation time of MMI training. In addition, some settings of $N$ yield a slight improvement compared with that when all possible classes are used. We achieve the best result of 95.98% when $N = 6$.

### 6.5.4 Comparison with the Other Classifier

To verify the efficiency of our proposed recognition system and the MMI training techniques, we compare their performance with the convolutional networks. The convolutional networks characterise a special instance of multi-layer networks trained by the gradient descent approach. They combine three ideas: i) local receptive fields; ii) shared weights; and iii) spatial sub-sampling, to ensure some degree of shift, scale and distortion invariants in the character samples. Since all the weights are optimised by back-propagation, the convolutional networks embed feature extractors in their classifier design through the use of multiple convolutional and subsampling layers. We compare our recognition system with a variation of the convolutional networks called *LeNet-5*, which is considered one of the most efficient classifiers [1]. The structure of *LeNet-5* used in the experiments is shown in Figure 6.13. Detailed descriptions of the optimisation process and the function of each layer can be found in [1].

We train the convolutional networks for 160 iterations, when no further improvement on both the training and testing sets is observed. The experiments on baseline and non-baseline characters are carried out separately, and recognition results are summarised in Table 6.6. The best MMI training results obtained from Section 6.5.3 are also displayed and categorised into two character groups. These results are slightly better than those of the convolutional networks. Clearly, it is shown that the performance of our proposed system and the MMI training techniques for the Thai handwriting recognition problems

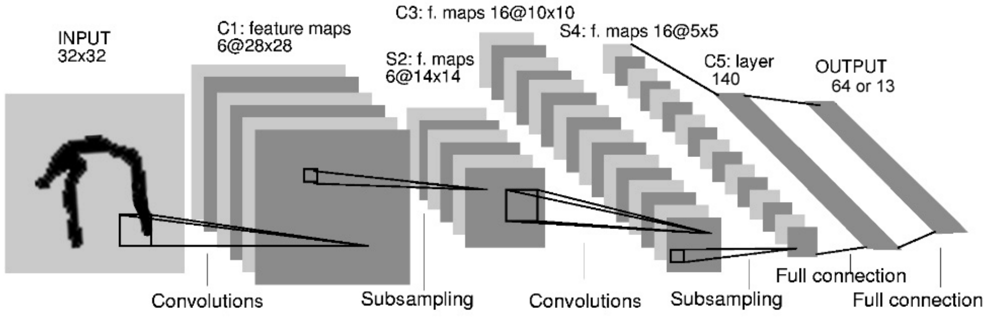


**Figure 6.13:** Structure of the convolutional networks, *LeNet-5*, used in the experiments, modified from [1]

| Thai Character Categories | Recognition Results | |
|---|---|---|
| | HMM-Based Classifier | Convolutional Networks Classifier (*LeNet-5*) |
| Baseline Characters | 95.92 % | 94.87 % |
| Non-Baseline Characters | 96.32 % | 95.92 % |
| Average | 95.98 % | 95.04 % |

**Table 6.6:** Comparison of the recognition performance between the proposed HMM-based classifier and the convolutional networks

lives up to the high performance achieved in other recognition systems.

Thus far, MMI training is carried out on two separate character groups by assuming that we have prior knowledge regarding the position (either baseline or non-baseline) of each character. The final experiments are conducted to perform MMI training under the circumstance that such information is error-prone or not accessible. In this case, all characters are treated similarly, and the recognition process compares the likelihoods among all possible 77 classes. The denominator likelihoods are then computed from the output of $N$-best recognition. The best result obtained from MMI training is 94.33% accuracy on the testing sets compared with 89.34% from ML training. This demonstrates how effective MMI training is.

## 6.6 Discussion and Summary

The problems of Thai handwriting recognition and discriminative training are ongoing research areas, and many of their aspects have not been explored. In this chapter, we present the use of discriminative training and propose novel feature extraction techniques to improve the performance of the HMM-based Thai handwriting recognition system. The best result we achieve is slightly better than the results obtained from the other high performance system. Several aspects of MMI training are also investigated. A number of contributions of this study can be summarised as follows:

- A database of Thai handwritten characters is constructed for this research. Because automatic Thai handwriting recognition is still in the early stages of research, most researchers are required to collect their own data. For the benefit of future studies, our database will be available to other researchers upon request.

- Various feature extraction techniques for the HMM-based recogniser are evaluated. These features are efficiently computed from the global distributions of shapes in the image structure and do not rely on a search for specially defined local structural features as often used in previous research. The sliding-window technique is proved effective for this purpose. We focus on three different feature extraction techniques, namely the pixel, moment-based and Gabor wavelet features. Among these three methods, the Gabor features yield the highest performance. This suggests that they are capable of capturing more of the underlying structures in the image while preserving their spatial relations. It is, thus, a desirable property for a feature extraction technique in the sliding-window approach.

- The block-based PCA technique and the use of composite images are proposed to improve the recognition results. The block-based PCA is an alternative way to apply PCA to the feature vectors that enhances the features extracted from vertical blocks in the frame image. It yields a slight increase in the recogniser performance compared to the PCA results. On the other hand, the composite image is a concatenation of an original image with a 90-degree rotated image and a polar transformation of itself. These methods result in a relative error reduction from 11% to 39%, depending on which other techniques are used.

- A variation of the Extended Baum-Welch algorithm proposed by Povey [10] for MMI training is successfully applied to our recognition problem, despite its initial purpose for continuous speech recognition applications. It demonstrates a fast convergence of the MMI objective function and the training set results rise quickly to 100%. However, the test set shows a more modest improvement, and an improvement from 91.40% in ML training to 92.67% in MMI training is obtained. This indicates a sign of overtraining in which the system is not well generalised to the unseen data.

- It is found that the probability scaling [73] can efficiently improve the generalisation of MMI training. The scaling factor applied to the log likelihoods enables more characters to compete with the most likely character. However, its value should be carefully chosen and is task- and data-dependent. We obtain the highest recognition accuracy of 95.74% which is equivalent to 50% relative error reduction compared with the ML result.

- The use of $N$-best lists allows us to compute the denominator likelihoods in a more efficient way and also demonstrates a slight improvement in the accuracies compared to when all possible classes are used. Rather than accumulating the probabilities from all possible classes, we take into account only the $N$ most likely candidates specified by the $N$-best lists. The best result we achieve is 95.98%. Clearly, the HMM parameters optimised by MMI training are empirically revealed to improve the recognition performance from the ML-trained systems.

- The evaluation of our database on the convolutional networks, or so-called *LeNet-*

*5*, known as one of the most efficient classifiers, is performed. It is shown that the performance of our proposed system and the MMI training techniques for the Thai handwriting recognition problems is slightly better than the high performance achieved in the *LeNet-5* system.

Although the database used in this research has not been widely used, we believe that MMI training is likely to give improvements in other HMM-based handwriting recognition systems. The use of composite images is an easy-to-implement technique which can be applied independently of other aspects of the recognition system, but only for isolated character recognition. However, more experiments are required to test whether the block-based PCA is of general use for feature processing. Further studies to evaluate these techniques on the common databases, e.g. NIST databases [141, 142], are also important. However, these are beyond the scope of this chapter and are left for future work. Finally, with an increasing number of HMM-based classifiers for large vocabulary cursive handwriting recognition, it is anticipated that the discriminative training techniques reported in this thesis may be of general use to improve the performance of those systems. Other future research perspectives are addressed in Section 9.2.

# Part III

# Discriminative Training in Prototype-Based Classifiers

Chapter 7

# Discriminative Prototype-Based Recognition Systems

## 7.1 Introduction

The main purpose of this chapter is to present the concept of discriminative training in prototype-based classifiers. The previous chapters in Part II extensively investigate the applications of discriminative training and HMM-based classifiers to the handwriting recognition problems. The HMM classifier is one of the *parametric* techniques which require an assumption for the correct form of the probability density function (pdf). Discriminative training in the HMMs demonstrates significant gain in recognition performance compared to ML training, even when the functional form of the underlying pdf may be incorrectly assumed. An alternative approach to pattern recognition is the *non-parametric* techniques that can be used without prior assumptions of the pdf's functional form. In this latter case, an entire training set is used as prototypic examples to infer the Bayes decision boundaries directly. However, this process is computationally expensive and requires a large storage space as the size of system parameters grows with respect to an increasing number of training samples. The *prototype-based classifier* attempts to reduce such computation requirements while simultaneously maintaining comparable classification performance. This can be performed either by selecting or generating subsets of the prototypes from the given set of training samples.

This chapter presents the concept of discriminative training, Minimum Classification Error (MCE) training in particular, in the prototype-based recognition systems. The concept of the non-parametric approach and its variations are introduced in Section 7.2 as a background for the prototype-based systems. We focus on the Prototype-Based Minimum Error Classifier (PBMEC) which employs the MCE criterion for prototype generation with the aim of achieving minimum classification errors. The formalisation of PBMEC and its training procedures are described in Section 7.3 as a foundation for its applications to handwriting recognition problems in Chapter 8. Recent applications of PBMEC and related classifiers are reviewed in Section 7.4. Finally, a summary of this chapter is given in Section 7.5.

## 7.2 Non-Parametric Classification

The primary goal of pattern recognition is to achieve minimum error rates. As discussed in Chapter 2, the minimisation of the error probabilities is guaranteed by the Bayes decision theory. Given an incoming pattern, it advocates choosing the category that yields the maximum posterior probability. However, the Bayes decision rule requires complete knowledge of all relevant probabilities which is rarely acquired in practice. We have to rely on an indirect method that estimates the probability densities from the training samples and uses them as if they are the true densities. Various approaches to density estimation can be categorised into two groups. The first one is the *parametric* techniques which require an assumption of the correct functional form of the underlying probability density. In this case, the parameters of the function are estimated by using the available training data. An example of the parametric techniques is the HMM-based classifier extensively discussed in Part II of the dissertation. Since the correct form of the distribution is rarely known, the chosen parametric function may not be a good representation of the true density. An alternative solution to this problem is the *non-parametric* approaches where the prior assumption for the pdf is not required, it is determined entirely by the data.

There are several non-parametric techniques. Some techniques, such as kernel-based methods, try to estimate the conditional density function $p(\mathbf{x}|C_j)$ from sample patterns without specifying the functional form in advance. The other approaches bypass the process of probability estimation and use the entire sample set to estimate the posterior probabilities $p(C_j|\mathbf{x})$. The classification decisions can be directly made by using these probabilities. An important instance of these techniques is the nearest-neighbour classifier. The formalisation of both methods are extensively described in [11, 12]. Section 7.2.1 presents an overview of the nearest-neighbour classifier and its variations, and the concept of prototype-based methods is discussed in Section 7.2.2.

### 7.2.1 Nearest-Neighbour Classifier

The nearest-neighbour (NN) classifier is one of the oldest and simplest classification methods widely used in various tasks, including data-mining, information retrieval and pattern recognition. Its popularity stems from a solid theoretical foundation known as the nearest-neighbour rule [143]. The basic principle of the NN classifier is based on an assumption that data of the same category generally exist within close proximity in the feature space and share similar properties. Hence, we could obtain its properties from the nearby samples, i.e. its *neighbours*. By applying this idea to pattern recognition, the category of the unseen sample can be inherited from the category of its nearest neighbour in the training set, as measured by the defined distance metric.

Consider a set of $N$ training samples $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ of $M$ categories $\{C_1, ..., C_M\}$. Each sample $\mathbf{x}_n$ is labelled with its corresponding category $\mathcal{C}^n \in \{C_1, ..., C_M\}$. The distance function, $dist(\cdot, \cdot)$, is a non-negative real-valued function defined for all sample pairs.

Sample $\mathbf{x}_i$ in the training set is regarded as the nearest neighbour of the unseen sample $\mathbf{x}$ when the condition, $dist(\mathbf{x}, \mathbf{x}_i) < dist(\mathbf{x}, \mathbf{x}_j)$ for all $i \neq j$, is satisfied. By using the NN rule, the category of $\mathbf{x}$ is inherited from $\mathbf{x}_i$, hence category $\mathcal{C}^i$. There are many choices of the distance metric which could be used. The commonly used one is the standard Euclidean metric although other metrics, like Mahalanobis distance, can also be applied.

The $k$-nearest-neighbour ($k$-NN) is an extension of the NN classifier. Rather than considering only the closest neighbour, the $k$-NN classifier takes into account the $k$ nearest samples around $\mathbf{x}$, as specified by the distance metric used. The label of $\mathbf{x}$ is inherited from the majority category among these $k$ samples. The value of $k$ is regarded as a smoothing parameter for the decision boundaries and is empirically determined. It can be seen that the NN (or 1-NN) classifier is a special case of the $k$-NN classifier where $k = 1$.

Despite its simplicity, the $k$-NN classifier has many advantages over other methods. For example, it requires no training process to classify the unseen samples and can be used with only a small training set. It is capable of adding new training samples during runtime. Furthermore, it is theoretically proven that the error rate of the $k$-NN classifier converges to the optimal error rate generated by the Bayes classifier when both the number of training data $N$ and the value of $k$ increase, and the ratio $k/N$ approaches zero [11]. This confirms that the $k$-NN classifier can asymptotically approximate the optimal classification decision. However, because all training samples have to be retained, the $k$-NN classifier suffers from a requirement for large storage space. There is also an associated computational cost since the distances between the testing sample and all training samples have to be computed. More sophisticated versions of the $k$-NN classifier are proposed in the literature [144, 145] to overcome these limitations for both storage efficiency and fast processing.

### 7.2.2 Prototype-Based Classifiers

One approach to deal with the disadvantages of the $k$-NN classifier is to find better representatives of the training samples rather than storing an entire training set. These representatives are referred to as the *prototypes* of the system, and a recogniser employing this technique is called a prototype-based classifier. It attempts to improve the storage requirement of the $k$-NN classifier and speed up the computation, while keeping comparable performance. There are two alternative approaches for the designing of prototype-based systems: *prototype-selecting* and *prototype-generating* approaches. In the prototype-selecting approach, a subset of the training samples is chosen by pruning the data based on some heuristic to maintain the performance of the system. In contrast, the prototype-generating scheme relies on the clustering algorithm and compresses the data into fewer representatives, based on some distortion measures. A comprehensive survey of these techniques can be found in [146]. Variations of these methods are briefly explained in the following paragraphs.

Many methods for prototype selection in the $k$-NN classifier have been proposed over the past thirty years. The selection process is concerned with finding some rules to determine which training samples are selected as the prototypes and which samples can be eliminated. Examples of the traditional techniques include the reduced NN, condensed NN and edited NN algorithms. Using the reduced NN rule [147], a sample is removed from the training set if the deletion does not result in misclassification of the remaining samples. The process is iteratively performed until no samples can be further deleted. On the contrary, the condensed NN approach [148] is considered as a bottom-up variation of the reduced NN rule. It starts with a prototype set containing one random training sample, and the misclassified training samples are added to the prototype set. The edited NN rule [149] retains the correctly classified samples in the prototype set. Extended versions of these approaches were proposed and evaluated on real problems [150]. The results demonstrated comparable classification performance to the traditional $k$-NN classifier.

On the other hand, the prototype-generating approach aims at iteratively constructing a modified training set to replace the original training samples. Methods for generating prototypes vary from simple unsupervised clustering to optimisation techniques that use heuristic error correction rules. The simplest approach is unsupervised clustering where the training samples of each category are clustered independently and the cluster centres then serve as the prototypes. Further modifications that employ supervised clustering and some optimisation techniques have also been proposed, such as Learning Vector Quantisation (LVQ) by Kohonen *et al.* [151, 152] and its extended versions. LVQ is capable of adjusting the prototypes so that the closest prototype of each training sample belongs to its correct category. Despite its popularity, the convergence of LVQ training is not guaranteed and it does not aim at optimising any objective function that has a formal link to the Bayes classifier. The prototypes generated by these conventional methods can be further reshaped by using the discriminative training criterion, as discussed Section 7.3.

## 7.3 Prototype-Based Minimum Error Classifier (PBMEC)

The Prototype-Based Minimum Error Classifier (PBMEC) is a prototype-generating classifier that employs discriminative training to create prototypes with the aim of minimising the classification error. It is unclear how the MMI criterion is applicable to the prototype-based system which is not explicitly probabilistic [153]. The MCE criterion is deemed more appropriate because it offers a general framework (with flexible choices for discriminant functions) that can be applied to the prototype-based system. In addition, the optimisation of the MCE objective function provides a direct link to minimising the probability of classification error and has a formal link to the designing of the Bayes classifier. PBMEC was originally proposed for speech recognition tasks by McDermott and Katagiri [38]. It appears to be a valid alternative to the standard prototype-based $k$-NN classifier in terms of performance, speed and memory requirement. The original version of PBMEC is used to classify variable-length patterns. It embeds the Dynamic Programming (DP) procedure in the same manner as the HMM classifier. However, the version of PBMEC that deals with fixed-length patterns proposed by Biem and Katagiri [154] is applied to this work. It requires only a single feature vector extracted from each character sample, as opposed to the sliding-window technique employed in the HMM recognition system. The underlying reason for using this system is that, given a limited time frame for this research, it provides a more feasible environment for comprehensive studies of certain important aspects of PBMEC that have never been addressed in the literature (see Section 7.4) than the one that uses variable-length patterns.

This section explains the details of the PBMEC recognition system, including its structure and training process. These involve the introduction of the discriminant function used in this work. We discuss the optimisation of the MCE objective function known as Generalised Probabilistic Descent (GPD) in detail. The gradient computations required by GPD optimisation and other gradient-based methods are also demonstrated.

## 7.3.1 Recogniser Structure

A distinct difference between PBMEC and the $k$-NN classifier is that, in the PBMEC system, the whole training set, as used in the $k$-NN classifier, is replaced by the prototypes generated by the discriminative training procedure. All prototypes belonging to the same classification category contribute to the class-membership decision via the use of a discriminant function. As discussed in Chapter 2, the discriminant function for each category needs to be defined, and the classification decision is made by choosing the category that results in the best discriminant value. By definition, the best value often refers to the highest discriminant value. However, the discriminant function used in PBMEC is a function of distances between the sample and the prototypes which indicates the degree of membership of the sample to a particular category. Hence, the smallest discriminant value represents the closest category. Although this may contradict the original definition of discriminant function, any maximisation problem can be reformulated as a minimisation problem and the resulting decision boundary is not affected. A formalisation of the PBMEC discriminant function based on [154] is given below.

We are given a finite set of $M$ categories $\{C_1, ..., C_M\}$. The model of the category $C_j$, $1 \leq j \leq M$, is represented by the parameter set $\lambda_j$ which is a set of $P_j$ prototypes $\{\mathbf{r}_{j,m} \,|\, 1 \leq m \leq P_j\}$. For simplicity, all categories are expected to have the same number of $P$ prototypes, i.e. $P_j = P$, for all $j$. The $m$-th prototype of category $C_j$ is denoted by $\mathbf{r}_{j,m}$. Each prototype is often referred to as a *reference vector*. The overall system parameters are denoted as $\Lambda = \{\lambda_1, ..., \lambda_M\}$. It is essential that the discriminant function in PBMEC is continuous and, at least, first-order differentiable with respect to the system parameters, $\Lambda$, because the estimation of $\Lambda$ is performed using gradient-based optimisation approaches.

Given a sample $\mathbf{x}$, the PBMEC discriminant function $g_j(\mathbf{x}; \Lambda)$ is a distance-based function that indicates the extent to which $\mathbf{x}$ belongs to the category $C_j$ and is defined as:

$$g_j(\mathbf{x}; \Lambda) = \left\{ \sum_{m=1}^{P} ||\mathbf{x} - \mathbf{r}_{j,m}||^{-\nu} \right\}^{-\frac{1}{\nu}}, \tag{7.1}$$

where $1 \leq \nu < \infty$. The discrimination function used in PBMEC is analogous to an $L_p$-norm [155] of distances between the sample $\mathbf{x}$ and the prototypes $\mathbf{r}_{j,m}$, except that a negative value of $p$ is used in this case. For later reference, we refer to this norm as the $L_\nu$-norm of distances. The term $|| \cdot ||$ in Eq. 7.1 denotes the distance function in which any suitable distance metric can be used. At this moment, we only focus on a simple Euclidean distance defined as:

$$\begin{aligned} ||\mathbf{x} - \mathbf{r}_{j,m}|| &= ((\mathbf{x} - \mathbf{r}_{j,m})^T(\mathbf{x} - \mathbf{r}_{j,m}))^{\frac{1}{2}} \\ &= \left( \sum_{d=1}^{D} (x_d - r_{j,m,d})^2 \right)^{\frac{1}{2}}, \end{aligned} \tag{7.2}$$

where both $\mathbf{x}$ and $\mathbf{r}_{j,m}$ are $D$-dimensional vectors, $\mathbf{x} = (x_1, ..., x_D)^T$ and $\mathbf{r}_{j,m} = (r_{j,m,1}, ..., r_{j,m,D})^T$. The use of Euclidean distance can be generalised to the Mahalanobis distance, as discussed Section 8.3.2.

Using the above discriminant function, the final classification is done by the simple classification rule:

$$\text{decide } \mathbf{x} \in C_i \quad \text{if} \quad g_i(\mathbf{x}; \Lambda) < g_j(\mathbf{x}; \Lambda) \quad \text{for all } i \neq j, \tag{7.3}$$

or alternatively:

$$\text{decide } \mathbf{x} \in C_i \quad \text{if} \quad i = \operatorname*{argmin}_j g_j(\mathbf{x}; \Lambda), \tag{7.4}$$

which chooses the category that yields the smallest value of the discriminant function. The $L_\nu$-norm distance in Eq. 7.1 shows that the contribution of each prototype to the discriminant function is modulated by the parameter $\nu$ with the property:

$$\lim_{\nu \to \infty} g_j(\mathbf{x}; \Lambda) = \min_m ||\mathbf{x} - \mathbf{r}_{j,m}||. \tag{7.5}$$

For a large value of $\nu$, the discriminant function converges to the distance between the sample and its closest prototype, meaning that the category of sample $\mathbf{x}$ is inherited from the closest prototype. This is the nearest-neighbor rule. Thus, the tuning of $\nu$ enables the emulation of various decision rules. This make PBMEC a generalisation of various classifier structures.

### 7.3.2 PBMEC Overview

By following the definition of the discriminant function in PBMEC, we focus on the training process that generates prototypes or reference vectors from the training samples. Figure 7.1 illustrates processes in PBMEC training and recognition stages. PBMEC training is performed in two steps: the process of prototype initialisation and the discriminative training process using the MCE criterion. The MCE training assumes a good pre-training configuration. Thus, the prototype initialisation is required to place the reference vectors in the proper general positions [17]. The MCE training process then re-adjusts these prototypes to achieve the minimum classification error status on the training set, and the trained prototypes are used for the recognition of unseen samples on the basis of the above classification rule.

The prototype initialisation process in PBMEC is typically done by the traditional $k$-means procedure. The $k$-means process is a clustering algorithm that performs on a category-by-category basis to generate a set of prototypes representing each category. Its

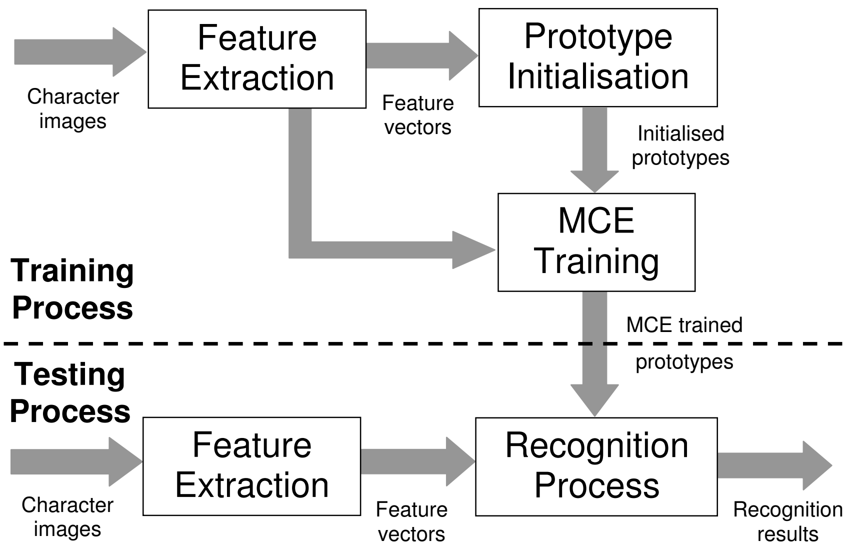


**Figure 7.1:** Process in PBMEC showing the detail of training and recognition stages.

general objective is to minimise the total distortion over the training data of the particular category.

Suppose the category $C_j$ consists of $N_j$ training samples and let $\{\mathbf{x}_1^j, \mathbf{x}_2^j, ..., \mathbf{x}_{N_j}^j\}$ be the training set of $C_j$. For a pre-chosen number of clusters $P$, the $k$-means algorithm iteratively adjusts the centre of each cluster, $\mathbf{r}_{j,m}$ where $1 \leq m \leq P$, by a within-cluster averaging process which minimises the distortion $E^{(j)}$ defined as:

$$E^{(j)} = \sum_{n=1}^{N_j} \min_m ||\mathbf{x}_n^j - \mathbf{r}_{j,m}||, \quad 1 \leq m \leq P. \tag{7.6}$$

The centres of resulting clusters serve as the initial prototypes to be further trained by MCE. An algorithm for $k$-means clustering iteratively performed on each category $C_j$ is given below [156]:

**Step 1:** Define the number of clusters, $P$, as a number of required prototypes and randomly choose $P$ samples, from the existing $N_j$ training samples, to be the cluster centres.

**Step 2:** Determine the closest cluster centre of each training sample and assign the sample to be a member of that cluster. The new clusters are subsequently formed.

**Step 3:** Obtain the centre of each new cluster by calculating the mean vector of the samples that belong to the cluster. Suppose cluster $m$ has $N_{j,m}$ samples, each sample can only be a member of one cluster, therefore $\sum_{m=1}^{P} N_{j,m} = N_j$. Let $\mathbf{x}_{n,m}^j$ be the $n$-th sample belonging to cluster $m$, the mean vector of cluster $m$ is computed by:

$$\mathbf{r}_{j,m} = \left(\frac{1}{N_{j,m}}\right) \sum_{n=1}^{N_{j,m}} \mathbf{x}_{n,m}^j \tag{7.7}$$

**Step 4:** Repeat Step 2 and 3 until the change in the distortion is less than the predefined threshold, i.e. there are only subtle changes in all clusters.

By following the above process, the optimal value of distortion in Eq. 7.6 can be achieved. The $k$-means clustering algorithm generates a disjoint set of clusters with well defined boundaries. This algorithm has been used as a standard prototype initialisation process in PBMEC until now. There is, however, an obvious discrepancy between the training and testing phrases in that the distance metric used in Eq. 7.6 does not conform with the $L_\nu$-norm distance used in the PBMEC discriminant function. In this work, we propose an alternative initialisation method which is better suited to the PBMEC system than the convention $k$-means algorithm. This method is introduced and evaluated in Section 8.3.3.

### 7.3.3 MCE Training in PBMEC

As discussed in Chapter 2, MCE training is a discriminative training approach that provides a direct relationship with the Bayes classifier. Applied to the prototype-based classifier, it allows the training process to directly focus on minimising the classification error. MCE training is performed by adjusting the initial prototypes after the prototype initialisation process. It aims at minimising the MCE objective function which is a reflection of the classification errors. Consider a set of $N$ training samples of $M$ categories, where category $C_j \in \{C_1, ..., C_M\}$ has $N_j$ training samples and $\sum_{j=1}^{M} N_j = N$. Let $\{\mathbf{x}_1^j, \mathbf{x}_2^j, ..., \mathbf{x}_{N_j}^j\}$ be the set of training samples of $C_j$ and let $\Lambda$ represent the overall system parameters. The MCE objective function, introduced in Eq. 2.22, can be rewritten here as:

$$f_{MCE}(\Lambda) = \frac{1}{N}\sum_{j=1}^{M}\sum_{n=1}^{N_j} \ell_j(\mathbf{x}_n^j; \Lambda), \tag{7.8}$$

where $\ell_j(\mathbf{x}; \Lambda)$ represents the loss incurred when the classification decision of the sample $\mathbf{x}$ is made, given that its correct category is $C_j$. Ideally, the step-wise 0–1 function should be used as the loss function. The loss is incurred when $\mathbf{x}$ is misclassified, i.e. $\mathbf{x}$ is recognised as $C_k$ where $k \neq j$. However, the optimisation of $f_{MCE}$ is usually performed by using gradient-based approaches which require a smooth loss function. The practical loss function is defined as a function of the misclassification measure, $d_j(\mathbf{x}; \Lambda)$, similar to Eq. 2.24:

$$\ell_j(\mathbf{x}; \Lambda) = \ell(d_j(\mathbf{x}; \Lambda)), \tag{7.9}$$

where $d_j(\mathbf{x}; \Lambda)$ is the misclassification measure of $\mathbf{x}$ given that its correct category is $C_j$. The important properties of the loss function, $\ell(d)$, can be summarised as follows [27]:

- Value of loss varies between 0 and 1.
- The loss is a monotonically increasing function.
- The loss limits are 0 and 1, for $d \to -\infty$ and $d \to \infty$, respectively.

There are several approximations in the literature that satisfy these conditions. The widely used ones are the sigmoid and piece-wise linear functions discussed in Section 2.4.2.

In this work, we use a more sophisticated loss function based on [27], which is described in Section 8.4.1.

The misclassification measure of $\mathbf{x}$ is a function that maps the classification decision of $\mathbf{x}$ into a scalar value. In addition, it provides an indication of how good the classification is. The misclassification measure of $\mathbf{x}$ is defined as a difference between the discriminant function for the correct category $C_j$, $g_j(\mathbf{x};\Lambda)$, and the discriminant functions of all other categories, $g_k(\mathbf{x};\Lambda)$, where $k \neq j$. The misclassification measure used in PBMEC is:

$$d_j(\mathbf{x};\Lambda) = g_j(\mathbf{x};\Lambda) - \left[ \frac{1}{M-1} \sum_{\substack{k=1 \\ k\neq j}}^{M} g_k(\mathbf{x};\Lambda)^{-\eta} \right]^{-\frac{1}{\eta}} . \tag{7.10}$$

This misclassification measure is slightly different from the one defined in Eq. 2.29, in particular, the sign of many of the terms is changed. This is because the discriminant function of PBMEC is a distance function whose smallest value represents the best category. However, both definitions result in similar behaviour in that a positive sign of the misclassification measure implies an incorrect decision, and a negative sign reflects a correct decision. Given the true category $C_j$, the correct decision for $\mathbf{x}$ occurs when $g_j(\mathbf{x};\Lambda)$ is the smallest discriminant value when compared to that of other categories. As a result, the second term in Eq. 7.10 is larger than the first term. This accordingly yields a misclassification measure with a negative sign. The $\eta$ parameter is a positive integer that controls the degree of contribution among the competing categories. It is an adjustable parameter that acts as a smoothing factor in the same manner as the $\nu$ parameter in the discriminant function. By setting $\eta$ to $\infty$, the misclassification measure is approximated by the difference between the discriminant value of the best, yet incorrect, category and that of the true category.

### 7.3.4 Generalised Probabilistic Descent (GPD) Optimisation for PBMEC

The MCE training in PBMEC is basically a process of parameter adjustments with an aim of minimising the MCE objective function. In PBMEC, the parameters of our concern are the prototypes. As discussed earlier, the typical methods for optimising the MCE objective function, $f_{MCE}$, are based on gradient-based approaches. In this section, the most commonly used method for MCE training, namely Generalised Probabilistic Descent (GPD) proposed by Katagiri *et al.* [28], is described. This optimisation technique is often referred to as MCE/GPD in speech recognition research. The GPD method is closely related to the Probabilistic Descent Method (PDM), one of the stochastic approximation methods, proposed by Amari [23]. A distinct difference is that GPD is a general and smoother version of PDM, and it can be applied to various recognition structures, including the recognition of variable-length patterns. An extensive discussion of GPD, together with its mathematical foundation, can be found in [28].

In fact, the MCE objective function in Eq. 7.8 is an average of the individual loss incurred for each training sample. Unlike the standard gradient descent methods, the

GPD optimisation focuses on the reduction of these individual losses rather than directly minimising $f_{MCE}$. The probabilistic descent theorem [23] proves that the minimisation of the individual losses leads to the minimisation of the average loss, at least to the local minimum. The adjustment of the system parameters in GPD is performed *adaptively* which means an update is done every time after the introduction of each training sample. This is opposed to the batch updating in the standard gradient descent methods in which parameters are adjusted after the entire training set is presented, i.e. after each training iteration. Let us assume that the sample $\mathbf{x}_\tau$ of category $C_j$ is presented at the discrete time index $\tau$. The parameter adaptation in GPD is performed according to:

$$\Lambda_{\tau+1} = \Lambda_\tau - \epsilon_\tau \mathbf{U} \nabla \ell_j(\mathbf{x}_\tau; \Lambda_\tau), \tag{7.11}$$

where $\Lambda_\tau$ represents the system parameters at the arrival time of $\mathbf{x}_\tau$, $\Lambda_{\tau+1}$ is the updated parameters and $\epsilon_\tau$ is a time-decreasing step size equivalent to a learning rate in gradient descent methods. Suppose we have an access to an infinite sequence of randomly selected training samples. According to the probabilistic descent theorem, the following conditions on $\epsilon_\tau$ are required for the optimisation to converge:

$$\sum_{\tau=1}^{\infty} \epsilon_\tau \to \infty \text{ and} \tag{7.12}$$

$$\sum_{\tau=1}^{\infty} \epsilon_\tau^2 < \infty. \tag{7.13}$$

However, only a limited number of training samples is available in practice. The learning rate $\epsilon_\tau$ is approximated by a monotonically decreasing function such as:

$$\epsilon_\tau = \epsilon_0 \left(1 - \frac{\tau}{N_{max}}\right), \tag{7.14}$$

where $\epsilon_0$ is the initial learning rate and $N_{max}$ is the preset total number of adjustment repetitions. In practice, training is repeatedly cycled over an entire set of training samples for a predefined number of iterations, $N_{Iter}$. This yields a total number of adjustments $N_{max} = N_{Iter} \times N$. The positive definite matrix $\mathbf{U}$ in Eq. 7.11 allows the learning rate for different model parameters to be scaled differently. The above adaptation procedure is applied to all adjustable PBMEC parameters. For the discriminant function in Eq. 7.1, the only parameters we are interested in are the reference vectors. Additional updatable parameters are introduced in Section 8.4.3. Clearly, the adaptation process involves the computation of the gradient of individual losses with respect to the prototypes. This process is described below.

### Gradient Computation

To compute the gradient of loss with respect to (w.r.t) the reference vectors, its partial derivatives w.r.t to each individual component of $\Lambda$ have to be found. These components consist of the misclassification measure, the discriminant function and the reference vectors. The final gradient can be computed by using the chain rule. Suppose we want to find the gradient of loss w.r.t the reference vector $\mathbf{r}_{a,p}$, which is the $p$-th prototype belonging to category $C_a$. We need to perform the derivative dimension-by-dimension. The $D$-dimensional vectors of sample $\mathbf{x}$ and the prototype $\mathbf{r}_{a,p}$ are denoted by $\mathbf{x} = (x_1, ..., x_D)^T$ and $\mathbf{r}_{a,p} = (r_{a,p,1}, ..., r_{a,p,D})^T$, respectively. According to the chain rule, the gradient of loss $\ell(d_j(\mathbf{x};\Lambda))$ of the sample $\mathbf{x} \in C_j$ w.r.t the $d$-th dimension element of such reference vector, $r_{a,p,d}$, is computed by:

$$\frac{\partial \ell(d_j(\mathbf{x};\Lambda))}{\partial r_{a,p,d}} = \frac{\partial \ell(d_j(\mathbf{x};\Lambda))}{\partial d_j(\mathbf{x};\Lambda)} \frac{\partial d_j(\mathbf{x};\Lambda)}{\partial g_a(\mathbf{x};\Lambda)} \frac{\partial g_a(\mathbf{x};\Lambda)}{\partial r_{a,p,d}}. \tag{7.15}$$

The derivative of loss $\ell(d_j(\mathbf{x};\Lambda))$ w.r.t the misclassification measure $d_j(\mathbf{x};\Lambda)$ is dependent on the loss function we choose. This derivative can be denoted by:

$$\frac{\partial \ell(d_j(\mathbf{x};\Lambda))}{\partial d_j(\mathbf{x};\Lambda)} = \ell'(d_j(\mathbf{x};\Lambda)). \tag{7.16}$$

The partial derivative of the misclassification measure $d_j(\mathbf{x};\Lambda)$ w.r.t the discriminant function $g_a(\mathbf{x};\Lambda)$ of category $C_a$ needs to be considered in two separate cases. The first case, $a = j$, occurs when the prototype belongs to the correct category of $\mathbf{x}$, and the second case when the prototype belongs to any other categories, $a \neq j$. These can be computed as follows:

**Case 1:** when $a = j$

$$\frac{\partial d_j(\mathbf{x};\Lambda)}{\partial g_a(\mathbf{x};\Lambda)} = 1 \tag{7.17}$$

**Case 2:** when $a \neq j$

$$\begin{aligned}
\frac{\partial d_j(\mathbf{x};\Lambda)}{\partial g_a(\mathbf{x};\Lambda)} &= \frac{-\partial}{\partial g_a(\mathbf{x};\Lambda)} \left[ \frac{1}{M-1} \sum_{\substack{k=1\\k\neq j}}^{M} g_k(\mathbf{x};\Lambda)^{-\eta} \right]^{-\frac{1}{\eta}} \\
&= \frac{1}{\eta} \left[ \frac{1}{M-1} \sum_{\substack{k=1\\k\neq j}}^{M} g_k(\mathbf{x};\Lambda)^{-\eta} \right]^{-\frac{1}{\eta}-1} \frac{\partial}{\partial g_a(\mathbf{x};\Lambda)} \left[ \frac{1}{M-1} \sum_{\substack{k=1\\k\neq j}}^{M} g_k(\mathbf{x};\Lambda)^{-\eta} \right] \\
&= \frac{1}{\eta} \left[ \frac{1}{M-1} \sum_{\substack{k=1\\k\neq j}}^{M} g_k(\mathbf{x};\Lambda)^{-\eta} \right]^{-\frac{1}{\eta}-1} \frac{1}{M-1} \frac{\partial g_a(\mathbf{x};\Lambda)^{-\eta}}{\partial g_a(\mathbf{x};\Lambda)}.
\end{aligned} \tag{7.18}$$

That is,

$$\frac{\partial d_j(\mathbf{x};\Lambda)}{\partial g_a(\mathbf{x};\Lambda)} = -\frac{g_a(\mathbf{x};\Lambda)^{-\eta-1}}{M-1} \left[ \frac{1}{M-1} \sum_{\substack{k=1\\k\neq j}}^{M} g_k(\mathbf{x};\Lambda)^{-\eta} \right]^{\frac{-1-\eta}{\eta}} \quad \text{when } a \neq j. \tag{7.19}$$

Finally, the partial derivative of the discriminant function $g_a(\mathbf{x};\Lambda)$ w.r.t the prototype element $r_{a,p,d}$ is performed by:

$$\begin{aligned}
\frac{\partial g_a(\mathbf{x};\Lambda)}{\partial r_{a,p,d}} &= \frac{\partial}{\partial r_{a,p,d}} \left\{ \sum_{m=1}^{P} ||\mathbf{x}-\mathbf{r}_{a,p}||^{-\nu} \right\}^{-\frac{1}{\nu}} \\
&= -\frac{1}{\nu} \left\{ \sum_{m=1}^{P} ||\mathbf{x}-\mathbf{r}_{a,p}||^{-\nu} \right\}^{-\frac{1}{\nu}-1} \frac{\partial}{\partial r_{a,p,d}} \left\{ \sum_{m=1}^{P} ||\mathbf{x}-\mathbf{r}_{a,m}||^{-\nu} \right\} \qquad (7.20)\\
&= -\frac{1}{\nu}\, ||\mathbf{x}-\mathbf{r}_{a,p}||^{-\nu-1} \left\{ \sum_{m=1}^{P} ||\mathbf{x}-\mathbf{r}_{a,m}||^{-\nu} \right\}^{\frac{-1-\nu}{\nu}} \frac{\partial ||\mathbf{x}-\mathbf{r}_{a,p}||}{\partial r_{a,p,d}}.
\end{aligned}$$

In the case of using the Euclidean distance, the last term of the above equation can be derived from Eq. 7.2 as:

$$\frac{\partial ||\mathbf{x}-\mathbf{r}_{a,p}||}{\partial r_{a,p,d}} = -\frac{(x_d - r_{a,p,d})}{||\mathbf{x}-\mathbf{r}_{a,p}||}. \qquad (7.21)$$

The final gradient is computed by substituting the above partial derivatives into Eq. 7.15. As mentioned, it is important to note that every function relating to the MCE training, including the loss function, is continuous and differentiable due to a requirement of gradient computations. Every time a new sample $\mathbf{x}$ is introduced, each reference vector of all categories is updated according to Eq. 7.11. From Eq. 7.17 and 7.19, it is noticeable that the update rules of two cases are different in their signs. This means the MCE/GPD update rule for PBMEC pulls the prototypes of the correct category closer to the input while the prototypes of other categories tend to be pushed away.

In fact, the MCE objective function in PBMEC can be minimised by using other optimisation algorithms, such as the conjugated gradient descent and the QuickProp algorithm described in Section 4.4.3. In these instances, the above gradient computation is accumulated, and the update process is performed after the introduction of all training samples, i.e. after one whole iteration.

After the MCE training process, the resulting prototypes can be used to recognise unseen samples as shown in the PBMEC testing process in Figure 7.1. Several experiments to evaluate the performance of PBMEC in handwritten character recognition problems are presented in Chapter 8. These include some important findings relating to PBMEC training.

## 7.4 Applications of Discriminative Prototype-Based Classifiers

PBMEC theoretically aims at achieving the ultimate goal of pattern recognition, which is minimum error, via the use of distance-based discriminant functions. However, it is debatable how well the discriminative prototype-based classifiers will perform in practical applications. Several efforts have been made to evaluate their performances over the past decade. The MCE training process, in common with other discriminative training techniques, involves more computation than non-discriminative approaches. Despite this fact, its computational requirement is outweighed by the compact and high-performance systems, as described below.

MCE training has been successfully applied to a prototype-based speech recognition system that employs Dynamic Programming (DP) for recognising sequential feature vectors. This recogniser is often known as the MCE/GPD trained Dynamic Time Warping (DTW) classifier. In [157], the system was evaluated on the phoneme recognition task by using the standard Bell Labs E-set database. It demonstrated better recognition results than the HMM classifiers trained with ML on the same data. Further experiments performed by Chang and Juang [158] increased the classification accuracies by using the exponential-form distance metric associated with adjustable weighting factors. Both methods required a large amount of computation for DP since they used full templates of phonemes for the representation of each speech category.

Instead of using full templates of phonemes, McDermott and Katagiri [38] applied PBMEC to model the speech category at the sub-phoneme level. The number of sub-phoneme states was relatively small compared to [158]. It resulted in a comparable recognition result while the amount of DP computation was significantly reduced. In later work by McDermott and Katagiri [159], the PBMEC system was extended to the task of continuous speech recognition by the use of a discriminant function that incorporated a finite state machine determined by the grammar of the task. A detailed explanation of the experiments in [38, 159] was summarised in [17]. Although an extensive evaluation of loss smoothing was presented in those reports, other important smoothing factors, such as $\nu$ in the discriminant function and $\eta$ in the misclassification measure, have never been addressed. Both $\nu$ and $\eta$ were set to infinity. The setting of these smoothing parameters needs to be studied in detail as it reflects the degree of contribution of competitive reference vectors and other competitive categories, respectively.

Despite its success in speech recognition applications, there have been few attempts in applying discriminative prototype-based classifiers to character recognition tasks. A successful application was reported by Huo *et al.* [44] where the classifier was applied to the task of printed Chinese character recognition. The system was trained on a very large database with more than three million samples of almost seven thousand character categories. The baseline result, using $k$-means clustering, on the testing set showed a high recognition accuracy of above 99%. After the MCE training, the absolute result was increased by 0.4%. Although this is equivalent to a 50% relative improvement in recognition performance, it would be interesting to evaluate the system on higher variance data, such as handwriting samples, where the initial accuracy is not high, and there is plenty of room for improvement.

Other important work was done by Liu *et al.* [146] in which eleven prototype learning algorithms, including MCE, were evaluated and compared on the handwritten digit and Chinese character recognition tasks. The results confirmed that the system trained with MCE provided the best recognition accuracy. The study also revealed that an improvement in the recognition performance of the testing set was not as good as that of the training set. This inability of a prototype-based classifier to generalise the unseen data is probably due to non-smooth decision boundaries generated by the nearest-neighbour rule. With this in mind, the effect of smoothing factors discussed above should be studied in detail.

## 7.5 Summary

The main purpose of this chapter is to present the concept of discriminative training in the prototype-based classifiers. An introduction to the non-parametric recognition approaches, which include the nearest-neighbour (NN) classifiers, provides the foundation of knowledge. The prototype-based classifier attempts to improve the storage requirement of the NN classifier and reduce its computational requirements, while maintaining comparable performance. Several methods proposed in the literature have been used to acquire the prototypes from the set of training samples. Among them, the method of using discriminative training to generate prototypes is promising since it aims to minimise classification errors. In this chapter, two important aspects are extensively discussed: the prototype-based classifier that employs MCE training, namely Prototype-Based Minimum Error Classifier (PBMEC), and its training process. Applications of PBMEC have demonstrated its advantages over the other methods. However, we found that several important aspects of PBMEC have never been addressed in the literature. Chapter 8 presents the applications of PBMEC to handwriting recognition problems and also takes into account these aspects.

# Chapter 8

# Evaluation of Discriminative Prototype-Based Classifiers

## 8.1 Introduction

The previous chapter described how discriminative training can be applied to the prototype-based recognition systems, MCE training in the Prototype-Based Minimum Error Classifier (PBMEC) in particular. PBMEC provides a compact and powerful classifier which is more applicable to the resource-constrained system than the conventional $k$-NN classifier. Although PBMEC demonstrates good performance on a large number of applications in speech recognition tasks, only a few efforts have been made to evaluate its performance on other applications. Despite its reliable efficiency, much effort is still required to improve the training process. For example, the PBMEC training process involves several settings of smoothing parameters. Due to an absence of guidelines for these settings, most implementations in the previous literature rely on an approximate method which somehow restricts the discrimination ability of the PBMEC system. Furthermore, an inconsistency between the criteria used in the conventional initialisation and the recognition processes may unnecessarily affect its recognition performance. For these reasons, the aim of this chapter is to experimentally evaluate the performance of PBMEC in handwriting recognition. We propose the methods to overcome the aforementioned problems and investigate various aspects of PBMEC training.

This chapter covers several key issues. First, the database and the feature extraction method used in the experiments are introduced in Section 8.2. All processes in PBMEC training are individually studied throughout the chapter. Section 8.3 focuses on the initialisation process and introduces the proposed initialisation method which is well suited to PBMEC. Rigorous studies and extensive evaluations of the MCE training process in PBMEC are performed in Section 8.4. These include a discussion of the loss function used in this work and the effect of smoothing parameters in MCE. The performance of PBMEC on different recognition tasks is compared with the baseline $k$-NN and other classifiers. Section 8.5 presents a brief introduction and the evaluation of the discrimina-

| Recognition Tasks | Numbers of samples | |
|---|---|---|
| | Training sets | Testing sets |
| Digit Task | 15,953 | 8,598 |
| Uppercase Task | 28,069 | 16,401 |
| Lowercase Task | 61,390 | 37,122 |

**Table 8.1:** The number of training and testing samples for each character recognition task in the UNIPEN database

tive feature extraction applications. Although the latter task is not the main objective of this work, it demonstrates an alternative application of MCE training applicable to feature transformation. Finally, the discussion and summary of the chapter are addressed in Section 8.6.

## 8.2 Database and Feature Extraction

Prior to evaluating any classifier, the database and the feature extraction method used in the experiments have to be defined. Among several publicly available databases, the international Unipen Foundation has provided researchers with a valuable handwritten character database, the UNIPEN database [160]. With a large number of handwriting samples contributed by universities and companies, the UNIPEN database offers a multi-writer handwriting database of English characters that consists of various character types, ranging from isolated characters to cursive handwritten words. Several reports have published recognition results based on this database and provided some reference points for the comparison of different classifiers [161]. The UNIPEN database is designed for the on-line handwriting recognition problems. The character is represented by its dynamic information, i.e. a sequence of points sampled at equally spaced time intervals. Nevertheless, in this work, the problem of interest is the recognition of isolated off-line handwritten characters, which requires the static information of the character data. With this in mind, the character images are generated from the given dynamic information by using the mapping technique described in [162]. The characters are binarised and size-normalised giving 22×30-pixel bi-level images. The training and testing sets used in the experiments are derived from the Train-R01/V07 and the DevTest-R01/V02 sets of the UNIPEN database, respectively. Three recognition tasks from subsets $1a$, $1b$ and $1c$ are considered, notably the digit recognition (10 classes), the lowercase character recognition (26 classes) and the uppercase character recognition (26 classes) tasks. Table 8.1 summarises the number of character samples for each recognition task.

The next process is to decide what features should be extracted from the character images. To achieve the highest recognition performance on any recognition task, it is necessary to undertake an extensive exploration of various feature extraction methods. However, due to a limited time frame for this research, only one feature extraction method is evaluated. Much effort is devoted to improving the PBMEC training process. Given a particular feature extraction method, the relative gains in recognition accuracy sufficiently indicate the effectiveness of this improvement.

As discussed Section 7.3, the architecture of PBMEC used in this research requires only a single feature vector for each character sample. One of the potential feature extraction methods is to use the spectrum-based or *spectral* feature,[1] whose value corresponds to a frequency component in the character image. Here, the two-dimensional Discrete Cosine Transform (DCT), which is the commonly used transform for image compression and pattern recognition [163], is used. The detailed computation of the DCT can be found in [164]. The DCT is more computationally efficient than the Discrete Fourier Transform (DFT) since it is based on the real part of the Fourier series, without complex variables involved. The strong energy compaction property of the DCT allows most of the signal energy of the image to be concentrated in a few coefficients of the DCT while the remaining coefficients are not significant. Therefore, parameter tuning for the DCT feature extraction can be performed by determining the number of coefficients required for the representation of the image sample. Once it has been transformed, the inverse DCT is used to reconstruct the image. The simplest indicator of measuring the difference between the original and the reconstructed images is the Root Means Square Error (RMSE).

The DCT is applied to the 22×30-pixel character image and accordingly generates the same size DCT image. Each pixel value in the DCT image corresponds to the DCT coefficient for each frequency. The most significant coefficients are located at the low frequency part, which is the top-left corner, of the DCT image. Low-pass filtering is performed on the DCT matrices to reduce the number of coefficients. By empirically varying the window size and reconstructing the images from the compressed DCT matrices, the number of coefficients necessary for the feature vector is obtained. Figure 8.1 illustrates the character samples and their reconstructed images based on different sizes of DCT images. The RMSE of each transformation is also shown. We choose the 4×6-pixel DCT image as the feature for the experiments throughout the chapter because it requires the smallest dimension of DCT whose reconstructed images are still recognisable by humans. This yields a 24-dimensional feature vector for each character image, which is equivalent to an approximate 70% compression (the feature vector uses only 30% of the storage of the 22×30-pixel image).

## 8.3 Evaluation of the Initialisation Process in PBMEC

In PBMEC, a prototype initialisation process is required to place the reference vectors into the proper generic positions. These prototypes offer a good pre-training configuration for the next MCE training stage. The following sections present the experiments to evaluate various aspects of the initialisation process in PBMEC and the proposed initialisation algorithm. These experiments are based on the database and feature vectors discussed in Section 8.2.

[1]The experiments in this chapter had been carried out before the effectiveness of the Gabor features presented in Chapter 6 was revealed.

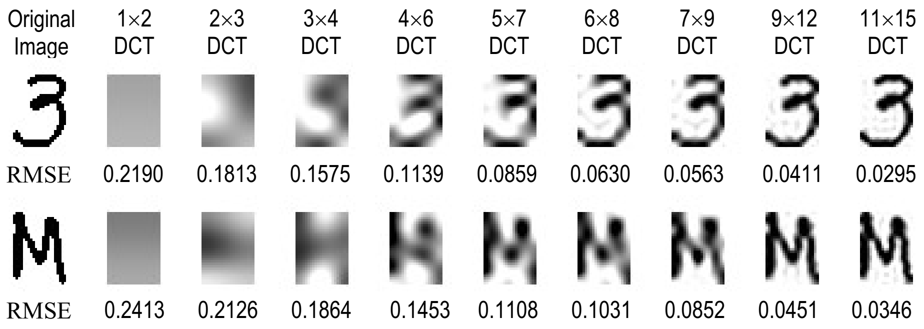

Original Image
1×2 DCT
2×3 DCT
3×4 DCT
4×6 DCT
5×7 DCT
6×8 DCT
7×9 DCT
9×12 DCT
11×15 DCT
RMSE 0.2190 0.1813 0.1575 0.1139 0.0859 0.0630 0.0563 0.0411 0.0295
RMSE 0.2413 0.2126 0.1864 0.1453 0.1108 0.1031 0.0852 0.0451 0.0346

**Figure 8.1:** Samples of the original character images and their reconstructed images based on various sizes of the DCT feature, with the corresponding Root Mean Square Error (RMSE) shown. The top and bottom rows represent the samples from digit "3" and uppercase character "M", respectively.

### 8.3.1 $L_\nu$-norm Contribution

The prototype initialisation process in PBMEC is usually done by the $k$-means clustering algorithm. The main objective is to generate a set of prototypes that minimises the total distortion over the training data, as a representative of each category. An algorithm for the $k$-means clustering was described in Section 7.3.2. After defining the number of clusters $P$, the training process is performed with the aim of minimising the distortion $E$, defined in Eq. 7.6, on a category-by-category basis. The results of this process are the reference vectors which serve as the initial prototypes to be refined by MCE training. However, it is interesting to evaluate the performance of these prototypes, prior to the MCE training process, by performing recognition on the unseen samples. At this stage, the overall training process of PBMEC is reduced from Figure 7.1 to the one illustrated in Figure 8.2.

Given the unseen sample, the recognition stage is performed by using the method described in Section 7.3.1 which computes the discriminant function for each category

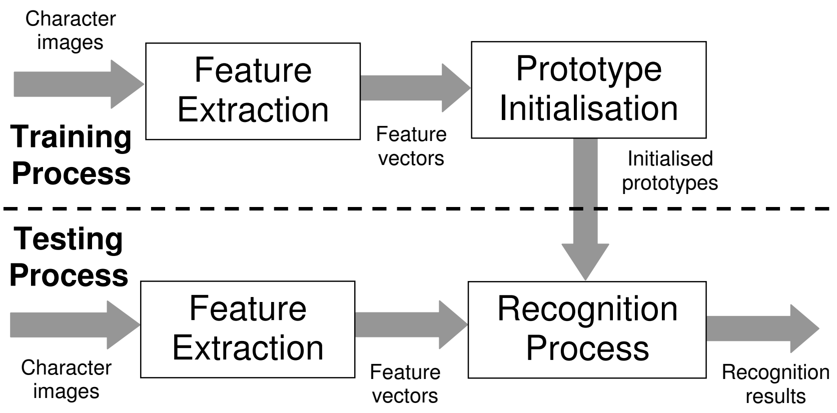


**Figure 8.2:** The reduced processes in PBMEC training for the evaluation of the prototype initialisation process

(Eq. 7.1). The category that yields the smallest value of the discriminant function is chosen as the recognised category of that sample. Note that the discriminant function involves the computation of an $L_\nu$-norm of the distances between the sample and all reference vectors. The setting of the parameter $\nu$ in the $L_\nu$-norm computation is regarded as the tuning of the smoothness degree in the decision boundaries. It controls the degree of contributions for each prototype over the discriminant function in the following manner: the higher the value of $\nu$ is, the less smooth the decision boundaries become. Non-smooth decision boundaries may result in a classifier that is less generalised to the unseen samples, as discussed in [26]. Moreover, the effect of the $\nu$ setting is carried on to the MCE training process, where the optimisation of its objective function directly involves the gradients of the discriminant function. From a literature survey in Section 7.4, it can be seen that most of the previous work tends to simplify the calculation of the discriminant function by setting $\nu$ to a very large value. This yields a discriminant function that converges to the distance between the sample and its closest prototype, which enables some computational savings. Thus far, there has been no theoretical guideline for choosing the appropriate degree of smoothing; therefore the only way is to empirically evaluate each setting.

We perform a series of experiments in order to study the effect of the $L_\nu$-norm contribution in the initialisation process. Each recognition task is performed separately. The distance metric employed in the experiments is the Euclidean distance. Since the number of prototypes for each category, $P$, has to be pre-specified, another set of experiments that empirically adjusts the value of $P$ is also carried out. In each setting of $P$, the system is trained by using the $k$-means algorithm and the samples from the training set. Then, in the recognition stage, the value of $\nu$ is varied and the classification performance on the testing set is measured.

Figure 8.3 shows the performance of various classifier settings in the digit recognition task. The horizontal axis represents the value of $\nu$ in the $L_\nu$-norm while the vertical axis denotes the recognition results on the testing set. Each graph corresponds to each setting of $P$, i.e. the number of prototypes per class. It is clear that the optimal performance is achieved when the $\nu$ value ranges from 2 to 4, and is independent of the number of prototypes per class. Note that setting $\nu$ to a large value is equivalent to using the nearest neighbour rule, these results suggest that the classical use of the nearest neighbour rule, whereby each category is represented by its closest prototypes to the sample, is not optimal. The degree of smoothing introduced by setting $\nu$ to a relatively small value in the $L_\nu$-norm of distances yields better performance. The results in both the lowercase and uppercase character recognition tasks also demonstrate similar trends. However, before this finding was revealed by Nopsuwanchai and Biem [165], we had already chosen the value of $\nu = 5$ as the initial setting for all experiments in the remaining sections. These experiments are extremely time consuming and re-running them with $\nu = 3$ or 4 will not give significantly different results. Although setting $\nu = 5$ does not result in the best performance, it is close to the best in all cases and provides a unique initialisation for all experiments which is valid for their comparisons. Table 8.2 summarises the best recognition results of all recognition tasks achieved when $\nu = 5$.

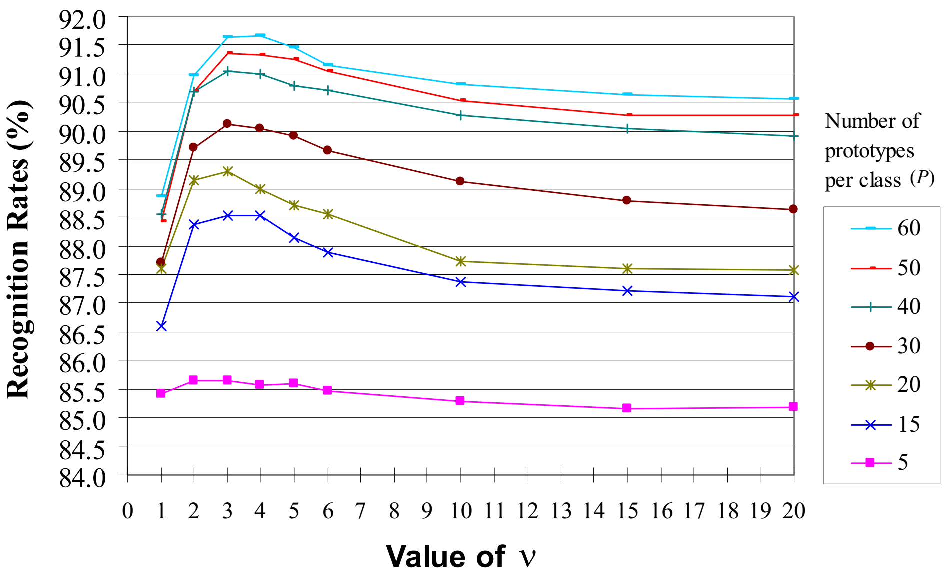


**Figure 8.3:** The recognition performance of the initialisation process in the digit recognition task showing the recognition results of the testing set versus the value of $\nu$ for various recognition structures

| Recognition Tasks | Initialised Recognition Results ($k$-means Clustering) |
|---|---|
| Digit Task | 91.45 % |
| Uppercase Task | 82.42 % |
| Lowercase Task | 73.37 % |

**Table 8.2:** The recognition performance of the $k$-means initialisation process in all recognition tasks

## 8.3.2 Effects of Distance Metrics

The distance metrics utilised in the previous experiments are based on the commonly used Euclidean distance metric. In fact, any suitable distance metric can be applied to both the $k$-means clustering process and the recognition process to compute the discriminant functions. Using the Euclidean distance means that all components (or dimensions) in the feature vector equally contribute to the calculation of its distance from the reference vectors. In this case, the surfaces of a constant distance correspond to the spheroids centred around the reference vectors. However, from the statistical point of view, a distance that involves the variability of each component is preferred. Components with high variability should receive less weight than those with low variability. This weighting property is obtained by scaling all components with their inverse of variances, which conforms to the definition of the Mahalanobis distance metric. The Mahalanobis distance

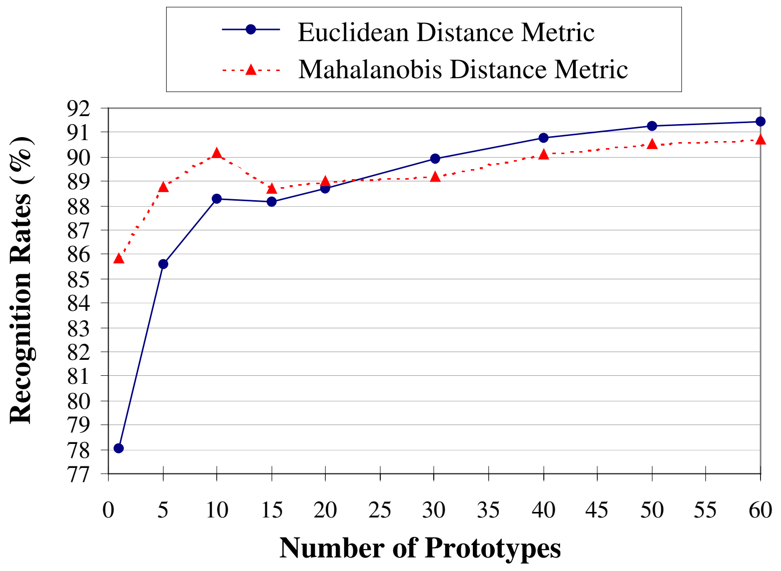


**Figure 8.4:** The recognition performance of the $k$-means initialisation process in the digit recognition task showing the recognition results of the testing set for different distance metrics ($\nu = 5$)

between the sample $\mathbf{x}$ and the prototype $\mathbf{r}_{j,m}$ is defined as:

$$\begin{aligned}\|\mathbf{x}-\mathbf{r}_{j,m}\|_{\mathrm{w}} &= ((\mathbf{x}-\mathbf{r}_{j,m})^T(\mathbf{\Sigma}_{j,m})^{-1}(\mathbf{x}-\mathbf{r}_{j,m}))^{\frac{1}{2}} \\ &= \left(\sum_{d=1}^{D}\frac{1}{\sigma^2_{j,m,d}}(x_d - r_{j,m,d})^2\right)^{\frac{1}{2}}, \end{aligned} \tag{8.1}$$

where $\mathbf{\Sigma}_{j,m}$ is the diagonal covariance matrix of the $m$-th cluster of the category $C_j$ and $\sigma^2_{j,m,d}$ is the corresponding variance of the $d$-th component. Using the Mahalanobis distance yields equally distanced ellipsoids whose axes are in the direction of the eigenvectors of $\mathbf{\Sigma}^{-1}$. For the special case where the variances in all components are identical, the Mahalanobis distance becomes equivalent to the Euclidean distance. Note that using the Mahalanobis distance yields twice as many adjustable parameters as using the Euclidean distance.

We perform the experiments to evaluate the performance of using the Mahalanobis distance metric in comparison with the Euclidean distance. In the $k$-means clustering process, the covariance matrices for all clusters and categories are computed to generate the weighting factors after each training iteration. The final covariance matrices are involved in the recognition process via the computation of distances in the discriminant functions. The recognition performance is compared with the results in Section 8.3.1. In the experiments, the value of $\nu$ is set to 5, and several settings of the prototype number per class are evaluated.

The recognition results from the testing set in the digit recognition task are shown in Figure 8.4. The horizontal axis represents the number of prototypes while the results of using two different distance metrics are shown in different graphs. From the results, it can be seen that the weighting factors introduced by the Mahalanobis distance are effective when the number of prototypes is small. For example, the result of 10 prototypes with the Mahalanobis distance is equivalent to the Euclidean distance with 30 prototypes. However, as the number of prototypes increases, the Euclidean distance metric yields better performance. The experimental results of the lowercase and uppercase character recognition tasks demonstrate similar behaviours. Although the underlying reason behind these results remains unexplained, it is probable that, when the number of prototypes increases, the Mahalanobis distance may require a large amount of training data to estimate the covariance matrices and may not have yet reached its optimal performance. A conclusion can be drawn from Figure 8.4 that, given a limited number of training samples, adding more prototypes asymptotically yields better results than introducing the weighting factors based on the covariance matrices.

Note that the weighting factors introduced to the PBMEC system serve as the supplementary adjustable parameters that can be refined by the MCE training, apart from the conventional prototype training. These additional parameter adjustments may enable further improvements in the MCE training stage, despite the degrading performance in the initialisation process. The MCE training experiments in Section 8.4.3 also takes into account this presumption.

### 8.3.3 $L_\nu$-norm-based $k$-means Initialisation

The $k$-means clustering algorithm, thus far, has been utilised as the standard initialisation process in PBMEC. It aims at generating a set of prototypes that minimises the total distortion over the training data. The distortion for each category is measured on the basis of the previously introduced Eq. 7.6. Clearly, the distortion measure in the $k$-means process uses the distance metric that does not match the $L_\nu$-norm metric used in the discriminant function (Eq. 7.1). The results in Section 8.3.1 confirm that different settings of $\nu$ in the $L_\nu$-norm metric provide significant variations in the recognition performance. The $k$-means clustering does not, however, take into account the setting of $\nu$. This creates a discrepancy between the $k$-means-based training and the recognition processes in PBMEC. With this in mind, we propose an alternative initialisation method, called the *$L_\nu$-$k$-means algorithm* [165], which is an $L_\nu$-norm-based $k$-means algorithm suitable for the PBMEC formalisation.

In the $L_\nu$-$k$-means algorithm, the distortion measure of the category $C_j$, encompassing $N_j$ training samples $\{\mathbf{x}_1^j, \mathbf{x}_2^j, ..., \mathbf{x}_{N_j}^j\}$, is defined as:

$$E_\nu^{(j)} = \sum_{n=1}^{N_j} \left\{ \sum_{m=1}^{P} ||\mathbf{x}_n^j - \mathbf{r}_{j,m}||^{-\nu} \right\}^{-\frac{1}{\nu}} . \tag{8.2}$$

This is equivalent to:

$$E_\nu^{(j)} = \sum_{n=1}^{N_j} g_j(\mathbf{x}_n^j; \Lambda), \tag{8.3}$$

where $\Lambda$ is the overall system parameters. The training process in the $L_\nu$-$k$-means algorithm aims to minimise the above distortion measures for each category by adjusting the reference vectors. Beginning with the randomly generated prototypes, the optimisation of Eq. 8.3 can be achieved by employing any gradient-based approaches derived from the gradient computation in Eq. 7.20. In the case of using the Mahalanobis distance, the last term in Eq. 7.20 becomes:

$$\frac{\partial \left\|\mathbf{x}-\mathbf{r}_{a,p}\right\|_{\mathrm{w}}}{\partial r_{a,p,d}} = -\frac{1}{\sigma_{a,p,d}^{2}} \frac{(x_d - r_{a,p,d})}{\left\|\mathbf{x}-\mathbf{r}_{a,p}\right\|_{\mathrm{w}}} \;, \tag{8.4}$$

and only the reference vectors are updated at each training iteration. The covariance matrices are computed after each iteration accordingly.

The distortion $E_{\nu}^{(j)}$ in Eq. 8.3 uses a metric similar to the one in the PBMEC's discriminant function and is a generalisation of the nearest-neighbour-based metric used in the classical $k$-means algorithm. It operates in a manner similar to the fuzzy $k$-means algorithms [166] as each input sample is not assigned to any single cluster defined by a single centroid. The distances to all centroids are taken into account in the computation of the distortion. It is clear from Eq. 8.2 that:

$$\lim_{\nu\to\infty} E_{\nu}^{(j)} = E^{(j)}, \tag{8.5}$$

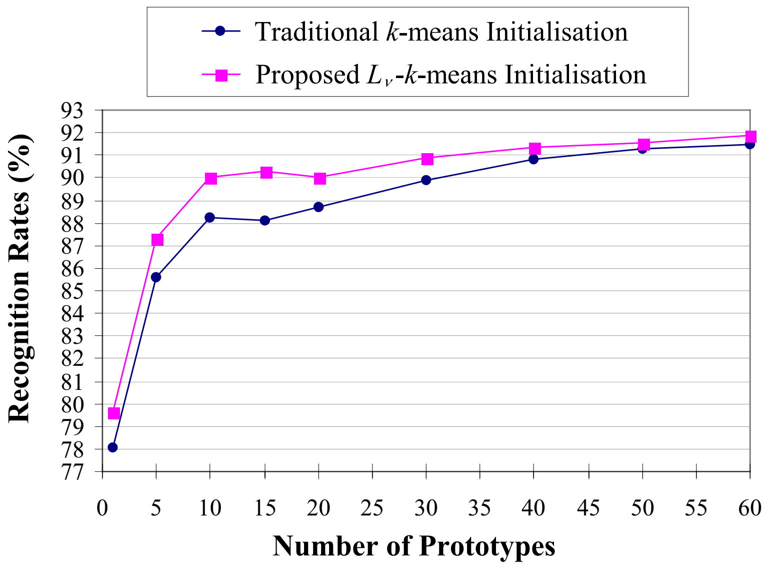
which means the $L_\nu$-$k$-means is equivalent to the classical $k$-means algorithm when the value of $\nu$ is large.

The performance of the proposed $L_\nu$-$k$-means is evaluated in the PBMEC initialisation process by using both the Euclidean and the Mahalanobis distance metrics. In the experiments, we set $\nu = 5$ and perform the QuickProp optimisation for 80 iterations. The recognition results are compared with the traditional $k$-means results in the previous sections. Figure 8.5 shows the performance of the test set versus the number of prototypes per class on the digit recognition task using (a) the Euclidean distance and (b) the Mahalanobis distance. These results show that, regardless of the distance metrics, the $L_\nu$-$k$-means is more efficient than the $k$-means algorithm, especially when a small number of prototypes are used. For example, in the context of the Euclidean distance, a 15% relative improvement from the $k$-means algorithm is achieved based on 15 prototypes per class but only a 5% relative improvement is gained when using 60 prototypes per class. Also, the $L_\nu$-$k$-means produces a more compact system: a system with 10 prototypes per class generated by the $L_\nu$-$k$-means algorithm yields similar performance to a system with 30 prototypes per class generated by the classical $k$-means algorithm. Similar results are also obtained in the lowercase and uppercase character recognition tasks. Table 8.3 summarises the best recognition results of the $L_\nu$-$k$-means algorithm in all recognition tasks when $\nu = 5$, compared with the results in Table 8.2. It shows that the $L_\nu$-$k$-means algorithm performs better than the classical $k$-means algorithm.

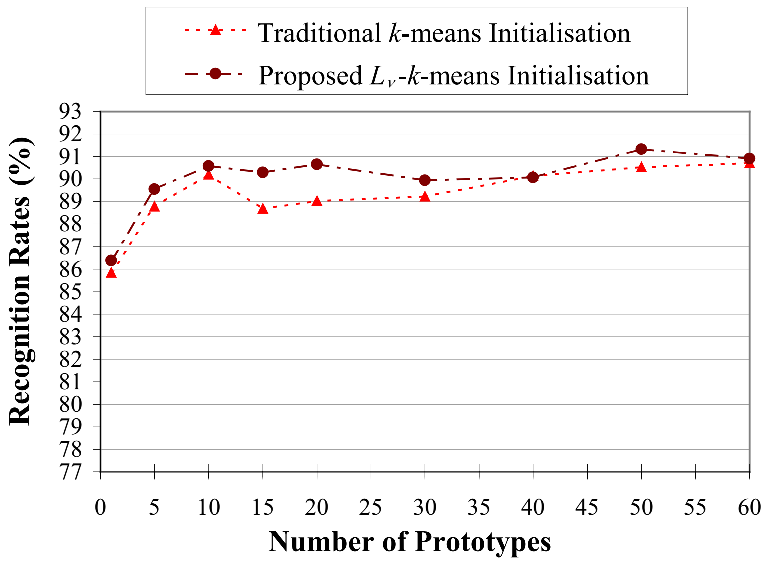
## 8.4 Evaluation of the MCE training process

### 8.4.1 The MCE Loss Function

In PBMEC, MCE training is performed by adjusting the initialised prototypes with the aim of minimising the average loss over the training samples. The essential aspect of



(a)



(b)

**Figure 8.5:** Comparison of the recognition performance between the standard $k$-means and the proposed $L_\nu$-$k$-means initialisation algorithms in the digit recognition task showing the recognition results on the testing set using (a) the Euclidean distance and (b) the Mahalanobis distance ($\nu = 5$)

| Recognition Tasks | Initialised Recognition Results | |
|---|---|---|
| | $k$-means Clustering | $L_\nu$-$k$-means Clustering |
| Digit Task | 91.45 % | 91.85 % |
| Uppercase Task | 82.42 % | 83.15 % |
| Lowercase Task | 73.37 % | 75.10 % |

**Table 8.3:** The recognition performance of the proposed $L_\nu$-$k$-means initialisation algorithm in all recognition tasks

MCE training is to define the loss function as a function of trainable parameters that reflects classification errors. In practice, the loss function is defined as a function of the misclassification measure converting it into the value between 0 and 1. The positive sign of the misclassification measure implies incorrect classification, hence incurring loss, while a negative sign means correct classification and yields no loss. As discussed in Section 2.4.2, the MCE approach approximates this ideal step-wise 0–1 loss function by a smooth and differentiable loss function which is more suitable to gradient-based optimisation. The majority of the MCE applications rely on either the sigmoid or the piece-wise linear function as their MCE loss function. Tuning the parameters $\alpha$ or $Q1$ and $Q2$ in Eq. 2.25 and 2.26 allows us to control the extent of learning for the ambiguous samples which appear close to the boundary between correct and incorrect classification. Although the use of these functions has demonstrated good performance during the optimisation process, the optimal choice for these parameters is difficult to determine. The work in [17] provides evidence that the setting of these values has a significant impact on the ability of the classifier to generalise the unseen data, and the optimal setting is achieved by empirically varying these parameters.

In this work, we utilise a more sophisticated loss function proposed by Biem [27], which automatically determines the loss parameters from the training data. Its underlying inspiration stems from the important properties of the loss function, $\ell(d)$, as discussed in Section 7.3.3, i.e.

- Value of loss varies between 0 and 1.
- The loss is a monotonically increasing function.
- The loss limits are 0 and 1, for $d \to -\infty$ and $d \to \infty$, respectively.

With the above definitions, the loss function can be viewed as the cumulative distribution function of the random variable $Y$ on the one-dimensional space spanned by the values of $d$. Its derivative is the probability density function of $Y$. Given the PBMEC system parameters $\Lambda$, the recogniser establishes a mapping between each sample $\mathbf{x}$ and the corresponding misclassification measure $d(\mathbf{x};\Lambda)$. We, therefore, can obtain the probability distribution of the misclassification measures $d$ for all samples. The advantage of considering the MCE loss as a distribution function in this way is that it can be automatically estimated from the data. By specifying the functional form of the probability density function, the loss parameters are estimated from the data based on the ML estimation using an EM algorithm. As a result, the loss $\ell(d)$ can be viewed as a probability measure

$P(Y < d; \Lambda)$ while its derivative $\ell'(d)$ is represented by $P(Y = d; \Lambda)$. Here, the derivative of the loss function is modelled by Gaussian mixtures, which are defined as:

$$\begin{aligned} \ell'(d) &= \sum_{i=1}^{L} c_i \, \frac{1}{\sigma_i \sqrt{2\pi}} \, \exp\left( - \frac{(d - \mu_i)^2}{2\sigma_i^2} \right) \\ &= \sum_{i=1}^{L} c_i \, \mathcal{N}(d, \mu_i, \sigma_i), \end{aligned} \tag{8.6}$$

where $L$ is a predefined number of mixtures, and the parameter $c_i$ controls the contribution of the $i$-th Gaussian mixture within the loss function. The $\mathcal{N}(\cdot, \mu_i, \sigma_i)$ is the normal distribution with the mean $\mu_i$ and variance $\sigma_i$ representing each mixture. From the definition of $\ell'(d)$ in Eq. 8.6, the corresponding loss function can be obtained by the anti-derivative function of $\ell'(d)$. Following is an example of the resulting loss functions which satisfies the MCE loss requirements:

$$\ell(d) = \sum_{i=1}^{L} c_i \left( \frac{1}{2} \, \mathrm{erf} \left( \frac{(d - \mu_i)^2}{2\sigma_i^2} \right) + \frac{1}{2} \right), \tag{8.7}$$

where $\mathrm{erf}(\cdot)$ is the error function (also called the Gauss error function) encountered in integrating the normal distribution, and is defined as:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-y^2) dy. \tag{8.8}$$

ML estimation is performed by applying the EM algorithm discussed in Section 3.3.1 directly [27, 48]. The optimal parameters of the loss function, which include $c_i$, $\mu_i$ and $\sigma_i$, are then obtained. The number of mixtures $L$ is the only parameter that needs to be pre-specified. It does not require as much rigorous tuning as the typical loss functions. The one-mixture case ($L = 1$) is analogous to using a sigmoid except that its parameters are learned from the data. Using more mixtures allows a modelling of asymmetric loss functions. Note that using extensive numbers of mixtures may generate a loss function that is overly fitted to the training data while being less generalised to the unseen samples. In this work, two Gaussian mixtures ($L = 2$) are used throughout the remaining experiments. The estimation of loss parameters is carried out as a separate process at every iteration of MCE training. A detailed explanation and illustration of this loss function can be found in [27].

### 8.4.2 Effects of Smoothing in MCE training

Smoothness in the decision boundaries is essential to the generation of a classifier that does not overly fit to the training samples, thereby providing good generalisation on the unseen data. In PBMEC, the degree of smoothness can be adjusted in many stages of the training process, such as the $L_\nu$-norm contribution in the initialisation process and the loss function tuning discussed in the previous section. In the MCE training process, another smoothing factor is also introduced. That is, the parameter $\eta$ in the computation

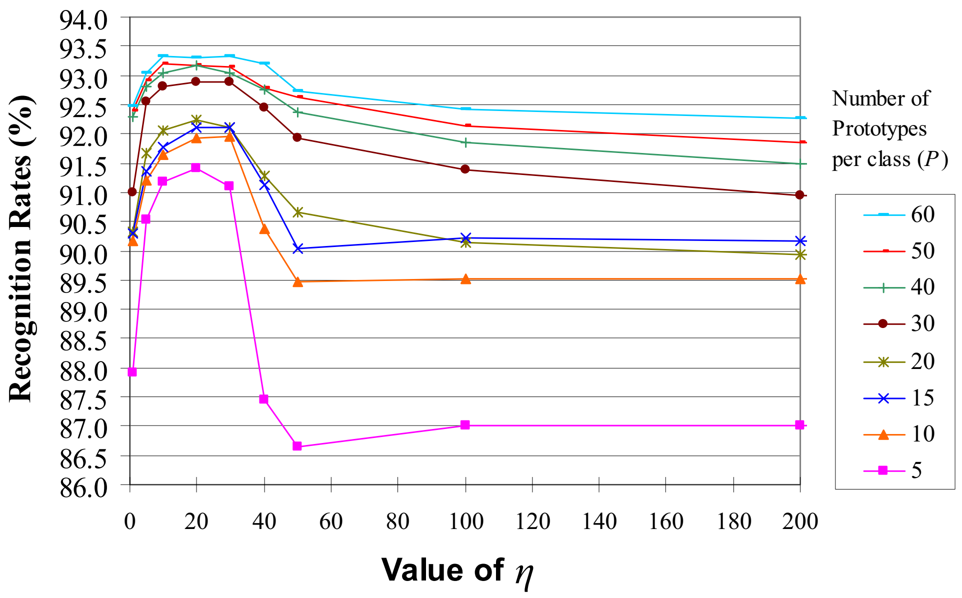


**Figure 8.6:** The recognition performance of the MCE training process in the digit recognition task showing the recognition results of the testing set versus the value of $\eta$ for various recognition structures.

of the misclassification measure in Eq. 7.10. The $\eta$ parameter is a positive adjustable parameter that controls the degree of contributions among the competing categories, hence the flexibility of the discriminative ability. It acts as a smoothing factor in the same manner as the $\nu$ parameter in the discrimination function. Most of the previous work relies on a simplification method that takes into account the correct category and the best competing category in particular. By specifying $\eta$ to be $\infty$, the misclassification measure is approximated to the difference between the discriminant value of the best, yet incorrect, category and that of the true category.

In this work, experiments are carried out to evaluate the effect of the $\eta$ parameter tuning in the same fashion as that of the $L_\nu$-norm contribution in Section 8.3.1. Starting with the $k$-means initialised prototypes, MCE training is performed by using the GPD optimisation and the previously discussed loss function. Parameter updating in MCE training is carried out for 80 iterations. We set the value of $\nu$ to 5 and vary the value of the $\eta$ parameter. The Euclidean distance metric is used in the experiments. The digit recognition performance for several classifier configurations on the testing set is shown in Figure 8.6. From the results, there exists a range of $\eta$ setting between 10 and 30 that provides a relatively high recognition performance. The experiments on the other two recognition tasks show similar results. These results suggest that the smoothness degree in the MCE training requires careful tuning, and that the very large value of $\eta$ widely used in the literature (see Section 7.4) does not produce optimal recognition performance.

### 8.4.3 Experimental Results of MCE Training in PBMEC

This section presents the experiments of evaluating the performance of MCE training in the PBMEC system. Typically, when the Euclidean distance is used to compute the discriminant function, the only parameters subject to training by MCE are the reference vectors. On the contrary, the use of the Mahalanobis distance metric introduces more adjustable parameters (i.e. the weighting factors) into the distance computation. Therefore, the goals of this section are to study the effect of using different distance metrics and to determine which system parameters contribute most to the highest recognition performance. Although the introduction of these parameters does not demonstrate good performance in the initialisation process, additional refinement by MCE training would enable further improvements. In this case, the options available for the MCE training are i) only reference vector training, ii) only weighting factor training, and iii) simultaneously training both reference vectors and weighting factors. The parameters that are not updated by MCE remain at their initialised values.

The weighting factors are trained by using the gradient-based optimisation approach in the same manner as in the reference vector training. The chain rule similar to Eq. 7.15 is applied to compute the gradient of loss with respect to the $d$-th dimension element of the weighting factor. Such an element of the $p$-th prototype belonging to category $C_a$ is represented by $w_{a,p,d} = \frac{1}{\sigma^2_{a,p,d}}$, hence a positive number. The training process, however, does not necessarily guarantee the final positive results. Therefore, the use of an auxiliary function is introduced to enforce this constraint on the training process. One applicable function is the log function, and the training process attempts to update the $\log w_{a,p,d}$ rather than the typical $w_{a,p,d}$. In this case, the following computation is required to complete the chain rule:

$$\begin{aligned} \frac{\partial \left\| \mathbf{x} - \mathbf{r}_{a,p} \right\|_{\mathrm{w}}}{\partial \log w_{a,p,d}} &= w_{a,p,d} \, \frac{\partial ||\mathbf{x} - \mathbf{r}_{a,p}||_{Mah}}{\partial w_{a,p,d}} \\ &= \frac{1}{2\,\sigma^2_{a,p,d}} \, \frac{(x_d - r_{a,p,d})^2}{\left\| \mathbf{x} - \mathbf{r}_{a,p} \right\|_{\mathrm{w}}}. \end{aligned} \tag{8.9}$$

Substituting Eq. 8.4 into Eq. 8.9 yields:

$$\frac{\partial \left\| \mathbf{x} - \mathbf{r}_{a,p} \right\|_{\mathrm{w}}}{\partial \log w_{a,p,d}} = -\frac{1}{2} \left( x_d - r_{a,p,d} \right) \frac{\partial \left\| \mathbf{x} - \mathbf{r}_{a,p} \right\|_{\mathrm{w}}}{\partial r_{a,p,d}}. \tag{8.10}$$

Given the loss function in Section 8.4.1 and the appropriate degree of smoothness (Section 8.4.2), the experiments of MCE training are performed by starting with the initial prototypes from the $L_\nu$-$k$-means clustering process. The QuickProp algorithm is utilised throughout the experiments as it empirically gives relatively higher improvements in the recognition performance than the GPD optimisation employed in the previous section. Parameter updating is carried out for 80 iterations, and the $\nu$ and $\eta$ parameters are set to 5 and 20, respectively. Figure 8.7 shows the recognition performance in the testing set of the digit recognition task for various training configurations. From the results, the MCE process can improve the relatively low performance generated by the Mahalanobis distance to the level equivalent to that of using the Euclidean metric. Even though,

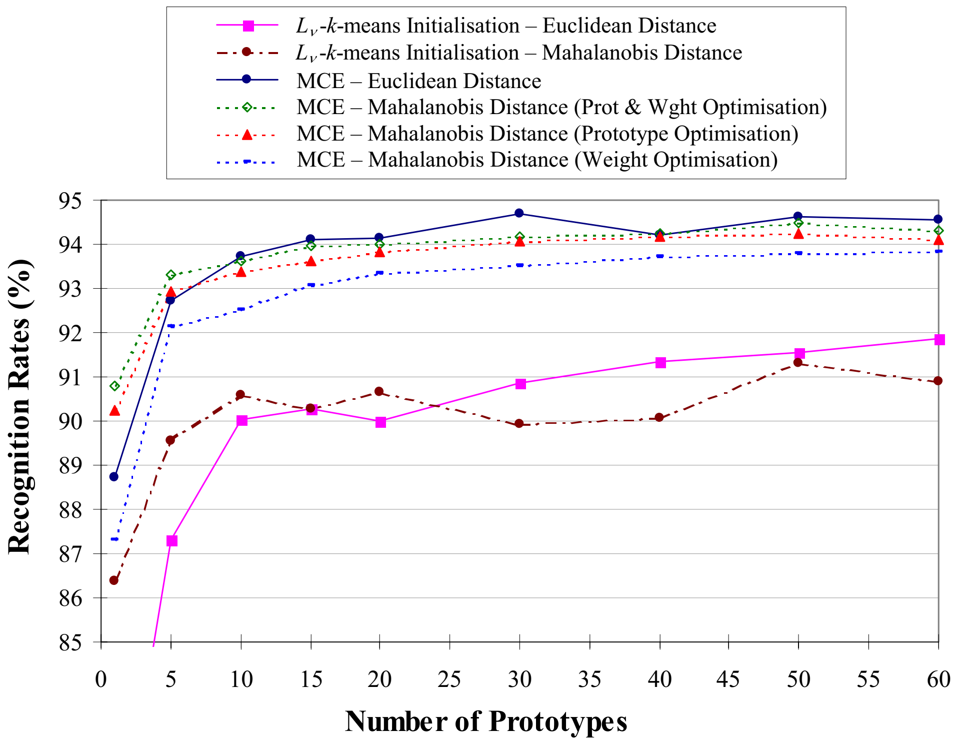


**Figure 8.7:** The recognition performance of the MCE training process in the digit recognition task showing the recognition results on the testing set using various distance metrics and various parameter training ($\nu = 5$ and $\eta = 20$)

in the case of using the Mahalanobis distance, prototype optimisation is more effective than weighting factor optimisation, the best result is achieved when both prototypes and weights are updated simultaneously. The best result obtained by MCE training gives the recognition rate at 94.69%, which is equivalent to a 42% relative error reduction from the $L_\nu$-$k$-means performance.

Experiments on the uppercase and lowercase recognition tasks were carried out. The best results achieved by MCE training among all distance metrics and all tasks are summarised in Table 8.4. Compared to the $L_\nu$-$k$-means results in Table 8.3, MCE training achieves a relative improvement between 15% and 35%, depending on the recognition task. It also displays the performance of the best $k$-NN classifier ($k = 5$) for each particular task and an estimated amount of storage required for each classifier. The amount of storage is approximated from the number of prototypes required to achieve the particular performance. The $k$-NN classifier experiments are based on the same feature extraction method used in the PBMEC system. It can be seen from the results that the performance of PBMEC and the $k$-NN are statistically equivalent. However, the PBMEC system requires significantly fewer parameters to reach the performance of the $k$-NN classifier. Although there are other applicable methods [137, 146] for the reduction of the storage space required in the $k$-NN classifiers, these results clearly demonstrate that the PBMEC training approach together with our proposed techniques yield compact recognisers which are suitable for deployment in limited-resource systems.

| Recognition Tasks | PBMEC System | | $k$-NN Classifier | |
|---|---|---|---|---|
| | Recognition Results | Sizes | Recognition Results | Sizes |
| Digit Task | 94.69 % | 28 kB | 94.39 % | 1.46 MB |
| Uppercase Task | 85.71 % | 195 kB | 85.90 % | 2.57 MB |
| Lowercase Task | 81.21 % | 195 kB | 81.30 % | 5.62 MB |

**Table 8.4:** Comparison of the recognition performance and the estimated amount of storage space between PBMEC and the $k$-NN classifier

| Recognition Tasks | OffSNT Classifier | | Convolutional Networks Classifier (*LeNet-5*) | |
|---|---|---|---|---|
| | Recognition Results | Sizes | Recognition Results | Sizes |
| Digit Task | 96.00 % | 6.1 MB | 95.87 % | 230 kB |
| Uppercase Task | 89.90 % | 59 MB | 88.90 % | 240 kB |
| Lowercase Task | 84.00 % | 76 MB | 85.22 % | 240 kB |

**Table 8.5:** Comparison of the recognition performance and the estimated amount of storage space between the off-line scanning $n$-tuple and the convolutional networks classifiers

Although this study does not aim at achieving the highest possible recognition performance on the UNIPEN database, it is interesting to compare the performance of the PBMEC system with other classifiers. The classifiers chosen for the comparison are the off-line scanning $n$-tuple (OffSNT) and the variation of the convolutional networks, namely *LeNet-5*, which are among the highly efficient classifiers. The OffSNT [162] extracts static features from the original character images to chain codes which are further segmented into several $n$-tuples, each of which becomes an index of the feature entry in the memory table. Despite its efficiency, the OffSNT requires an extensive amount of storage spaces since all related $n$-tuple templates need to be located. On the other hand, the convolutional networks [1] characterise a special instance of multi-layer networks trained with the gradient descent approach. Its distinct property is the feature extractor which is embedded in the classifier design through the use of multiple convolutional layers (see Section 6.5.4). The performance comparison of these classifiers for each recognition task[2] is shown in Table 8.5. Compared with the results in Table 8.4, these classifiers exhibit better recognition performance than the PBMEC system. The underlying reason behind these results is mainly because the features used in PBMEC are less efficient than the other two classifiers. In fact, it is unfair to compare their performance directly since the input of both the OffSNT and *LeNet-5* is based on the bitmap images, which conveys whole information of the characters to the classifiers, whereas our PBMEC implementation relies on the compressed (only 24 dimensions) Discrete Cosine Transform (DCT)

[2]The OffSNT results are courtesy of Eugene Ratzlaff, IBM T. J. Watson Research Center.

feature vectors. Obviously, an in-depth study is required to select more powerful feature extraction methods for PBMEC, and such study is beyond the scope of this dissertation. Nevertheless, the results suggest that PBMEC, together with our proposed techniques, has the potential to afford a compact classifier with high performance equivalent to other efficient classifiers.

## 8.5 Discriminative Feature Extraction in PBMEC

Feature extraction is considered one of the most important stages in classification problems. It is a process of extracting essential features from each input pattern while the classification process attempts to assign a class label to the particular features. Most conventional feature extraction methods are constructed in isolation from the design of classifiers. The criteria used in the feature extraction process are not consistent with those used in the classification process, hence there is no interaction between the design of these two stages. Although the conventional methods have demonstrated successful results to some extent, there is plenty of room for improvement. Several feature transformation techniques can be applied to reduce dimensionality of the extracted features and to generate a de-correlated set of features which may or may not take into account the discrimination between different classes. Examples of such transformation techniques are Principal Component Analysis (PCA, see Section 6.3.2) and Linear Discriminant Analysis (LDA) [11]. Nevertheless, these transformations are not optimised in accordance with the training criteria of the classifiers. Once transformed, the features also remain unchanged throughout the training process. The results from Section 8.4.3 suggest that the high performance of the convolutional neural networks stems from its efficient feature extractor which is embedded in the classifier design, and the optimisation of both the feature extractor and the classifier is performed simultaneously.

Thus far, this chapter has featured the applications of discriminative training in the classification process. However, discriminative training is also applicable to feature extraction, thereby providing a unified framework that utilises the same training criteria for both feature extraction and classification processes. The discriminative feature extraction (DFE) approach proposed by Biem and Katagiri [154, 167, 168] provides a novel method to classifier design. The purpose of DFE is to generate the optimal feature extractor as well as the classifier by using the objective function that is directly related to the classification error. DFE is an extended application of the MCE/GPD discriminative training to feature optimisation which is embedded in the design of an MCE-trained classifier. It has been successfully applied to many speech recognition tasks and has demonstrated an improvement in the recognition performance. Detailed summaries of these applications can be found in [27].

In the light of this, the potential applications of discriminative training to the feature extraction process in the PBMEC context are briefly introduced. Its objective is to incorporate DFE into the PBMEC framework based on [167] and to evaluate the DFE performance in the character recognition problems. The diagram in Figure 8.8 shows the modified processes in the PBMEC system where the additional feature transformation process is presented. The classifier and the feature transformer are jointly optimised by

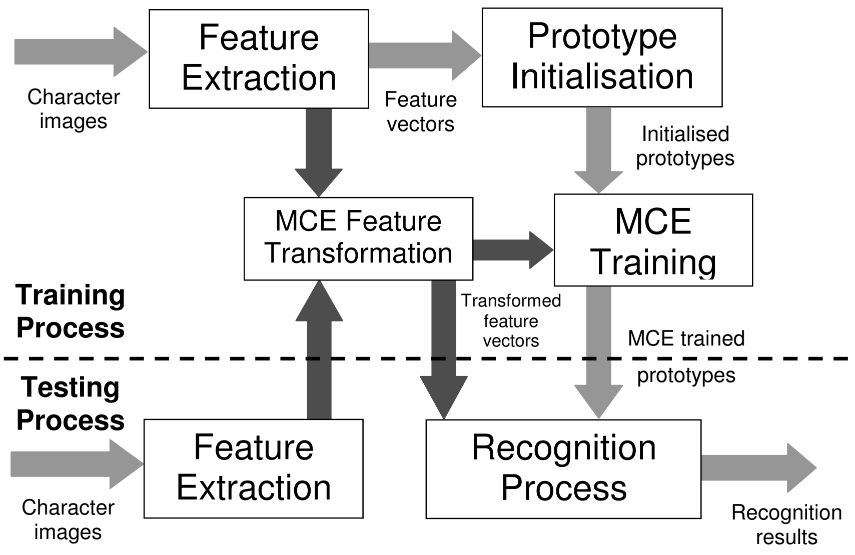


**Figure 8.8:** Process of DFE training in the PBMEC framework showing the additional feature transformation process

using the MCE criterion. The formalisation of this process and its evaluation are discussed in the following sections.

### 8.5.1 Formalisation of DFE in PBMEC

The feature extraction process can be viewed as a mapping from the input sample $\mathbf{s}$ to a corresponding feature pattern $\mathbf{x}$ through a function $\mathcal{F}(\cdot)$, given its parameter set $\Theta$, i.e. $\mathcal{F}_\Theta(\mathbf{s}) = \mathbf{x}$. The discriminant function $g_j(\mathbf{x}; \Lambda)$, therefore, is represented by $g_j(\mathcal{F}_\Theta(\mathbf{s}); \Lambda)$. It is clear that the feature extraction and classification processes are constructed based on different sets of parameters, $\Theta$ and $\Lambda$ respectively, whose optimisation is performed separately. The DFE approach attempts to integrate the design of these processes by using a single set of system parameters $\Phi = \{\Theta, \Lambda\}$. The resulting discriminant function becomes $g_j(\mathcal{F}_\Phi(\mathbf{s}); \Phi)$ and subsequently allows the feature extractor to participate in the classification decision. Obviously, there are various choices for the feature extraction function $\mathcal{F}_\Phi(\cdot)$, ranging from the linear to non-linear mapping. As a preliminary study, our focus is on the linear transformation function which is a product of the transformation matrix $\mathbf{t}$ and the input sample $\mathbf{s}$, i.e. $\mathbf{x} = \mathbf{ts}$.

Let $\mathbf{x}$ be the $D$-dimensional feature vector $(x_1, ..., x_D)^T$, and $\mathbf{s}$ be the $L$-dimensional input sample $(s_1, ..., s_L)^T$, the $d$-th element of $\mathbf{x}$ is computed by:

$$x_d = \sum_{l=1}^{L} t_{d,l}\, s_l, \tag{8.11}$$

where $t_{d,l}$ is the element in the $d$-th row and $l$-th column of the transformation matrix $\mathbf{t}$. MCE training in DFE aims at finding the transformation matrix that minimises either the MCE objective function or the individual loss. The gradient-based optimisation methods, such as GPD or QuickProp, can be applied for this purpose. This requires a gradient computation of loss with respect to each individual component of the transformation matrix. According to the chain rule and the misclassification measure definition (Eq. 7.10), the gradient of loss $\ell(d_j(\mathbf{x};\Phi))$ of the feature vector $\mathbf{x} \in C_j$ w.r.t the transformation matrix element $t_{d,l}$ is derived as:

$$\frac{\partial \ell(d_j(\mathbf{x};\Phi))}{\partial t_{d,l}} = \ell'(d_j(\mathbf{x};\Phi)\left(\frac{\partial g_j(\mathbf{x};\Phi)}{\partial t_{d,l}} - \frac{\partial G_j(\mathbf{x};\Phi)}{\partial t_{d,l}}\right), \tag{8.12}$$

where $G_j(\mathbf{x};\Phi)$ is the second term in Eq. 7.10, i.e.

$$G_j(\mathbf{x};\Phi) = \left[\frac{1}{M-1}\sum_{\substack{k=1\\k\neq j}}^{M} g_k(\mathbf{x};\Phi)^{-\eta}\right]^{-\frac{1}{\eta}}. \tag{8.13}$$

The computation of $\frac{\partial g_j(\mathbf{x};\Phi)}{\partial t_{d,l}}$ in Eq. 8.12 is performed by the following chain rule:

$$\frac{\partial g_j(\mathbf{x};\Phi)}{\partial t_{d,l}} = \frac{\partial g_j(\mathbf{x};\Phi)}{\partial x_d}\frac{\partial x_d}{\partial t_{d,l}}. \tag{8.14}$$

Consider Eq. 7.1 and 8.11, when applying DFE to the PBMEC system, the above partial derivative becomes:

$$\frac{\partial g_j(\mathbf{x};\Phi)}{\partial t_{d,l}} = \sum_{m=1}^{P}\left(\frac{-\partial g_j(\mathbf{x};\Phi)}{\partial r_{j,m,d}}\right)s_l. \tag{8.15}$$

By multiplying $\ell'(d_j(\mathbf{x};\Phi))$ to the above equation, this gradient can be computed based on the gradient of loss with respect to the prototype in Eq. 7.15:

$$\ell'(d_j(\mathbf{x};\Phi))\frac{\partial g_j(\mathbf{x};\Phi)}{\partial t_{d,l}} = -\sum_{m=1}^{P}\frac{\partial \ell(d_j(\mathbf{x};\Phi))}{\partial r_{j,m,d}}s_l. \tag{8.16}$$

In the same manner, the second term in Eq. 8.12 is derived as:

$$\ell'(d_j(\mathbf{x};\Phi))\frac{\partial G_j(\mathbf{x};\Phi)}{\partial t_{d,l}} = -\sum_{\substack{k=1\\k\neq j}}^{M}\sum_{m=1}^{P}\frac{\partial \ell(d_j(\mathbf{x};\Phi))}{\partial r_{k,m,d}}s_l. \tag{8.17}$$

Finally, by substituting Eq. 8.16 and 8.17, the gradient computation in Eq. 8.12 is obtained by:

$$\frac{\partial \ell(d_j(\mathbf{x};\Phi))}{\partial t_{d,l}} = -\sum_{k=1}^{M}\sum_{m=1}^{P}\frac{\partial \ell(d_j(\mathbf{x};\Phi))}{\partial r_{k,m,d}}s_l. \tag{8.18}$$

The gradient computation in the above equation is rather intuitive since the transformation matrix is shared among all samples of each category, hence summation across all categories. From the above equation, the discriminative ability of this gradient inherits from the gradient of loss with respect to the prototypes, which is embedded in its computation. The optimisation of both the transformation matrix and the prototypes is performed simultaneously, leading to a unified framework for feature extraction and classification in the PBMEC.

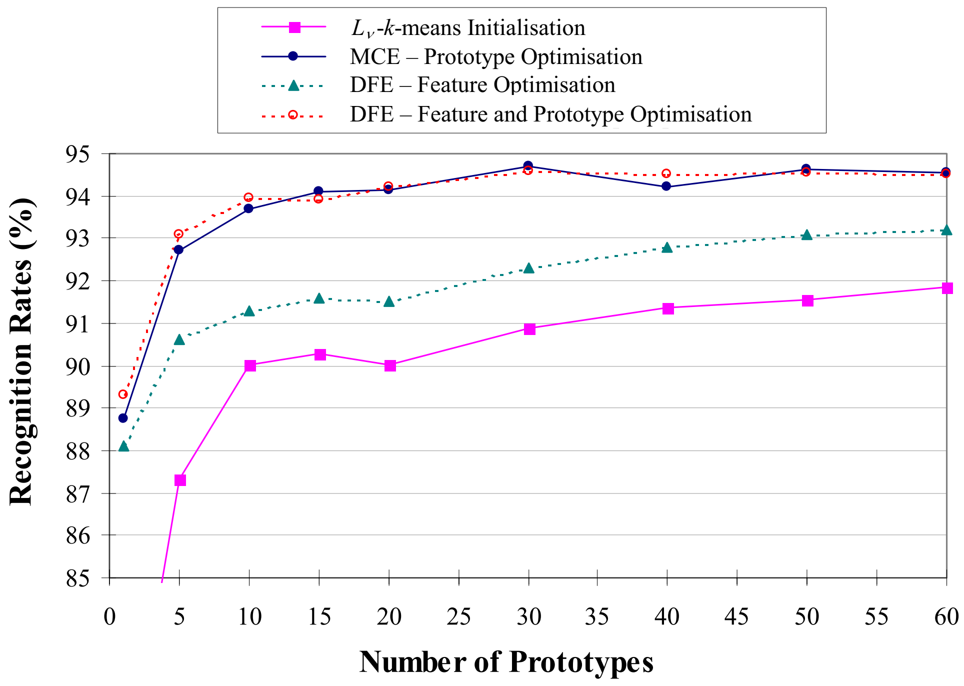


**Figure 8.9:** The recognition performance of the DFE training in the digit recognition task showing the recognition results on the testing set using various parameter training ($\nu = 5$ and $\eta = 20$)

## 8.5.2 Experimental Results of DFE in PBMEC

This section is devoted to the evaluation of DFE in the PBMEC system, based on its formalisation in Section 8.5.1. The DFE experiments, which utilise the Euclidean distance metric, are carried out in the digit recognition task. DFE is applied to map the input $\mathbf{s}$, the 24-dimensional DCT vector, to the DFE-transformed vector $\mathbf{x}$ that has similar dimensionality to $\mathbf{s}$. In this case, the transformation matrix $\mathbf{t}$ is the $24 \times 24$ matrix subject to optimisation by MCE. Beginning with the initial prototypes from the $L_\nu$-$k$-means clustering process and the transformation matrix $\mathbf{t}$ initialised by the identity matrix, the DFE training process simultaneously adjusts both the prototypes and the transformation matrix based on the MCE criterion. The results are then compared with the MCE training results in Section 8.4.3. An alternative training condition is obtained by optimising only the transformation matrix by DFE while leaving all prototypes at their initial positions. The advantage of this latter context is that it enables an exclusive evaluation of DFE without the contribution of the prototype training.

In the experiments, the $\nu$ and $\eta$ parameters are set to 5 and 20, respectively. The DFE training is carried out for 80 iterations using the QuickProp algorithm. Figure 8.9 shows the recognition results for various system configurations. It is interesting to note that even though all samples and categories share a single feature transformation matrix, its optimisation alone yields around 15% to 25% error reduction from the initial configuration, which is significant, given the fact that all prototypes are fixed at their initial positions. However, compared with the MCE results, the prototype optimisation is far more efficient than the feature optimisation. Furthermore, the joint optimisation of features and prototypes is expected to produce the best recognition accuracy, yet it only achieves performance equivalent to the prototype optimisation results. Our assumption is that the system may have reached its optimal results, hence indicating the highest capability obtainable from the compressed 24-dimensional DCT features. The reduction of the DCT coefficients could drop a significant amount of information from the original bitmap image, as discussed in Section 8.2. For a more comprehensive study of DFE, the transformation process should use the original bitmap image (rather than the compressed 24-dimensional DCT features) as the input sample $\mathbf{s}$ and allow the transformation matrix optimised by DFE to generate the transformed features based on the MCE criterion. Another alternative is to use a non-linear feature transformation instead of the linear transformation discussed in this section. However, these studies are beyond the scope of this dissertation.

## 8.6 Discussion and Summary

The Prototype-Based Minimum Error Classifier (PBMEC) has demonstrated good performance on a large number of applications in speech recognition tasks. It employs the MCE criterion to create prototypes with the aim of minimising the classification error. In this chapter, the successful applications of PBMEC to the off-line handwriting recognition problems are described. Despite its reliable efficiency, much effort is still required to improve the training processes of PBMEC. With this in mind, each training process in PBMEC is extensively studied. The data set used for the evaluation is based on the UNIPEN database. The key findings of this chapter can be summarised as follows:

- In the initialisation process, the evaluation of an $L_\nu$-norm contribution to the PBMEC discriminant function is conducted. The setting of the $\nu$ parameter in the $L_\nu$-norm is regarded as the tuning of the smoothness degree in the decision boundaries, which has a significant influence over the recognition performance. Although widely used in the majority of the previous work, an approximation method that specifies $\nu$ to be a very large value is proven not optimal.
- The PBMEC discriminant function involves the computation of distances between the samples and the prototypes to which any suitable distance metrics can be applied. The effects of using two different distance metrics are examined. It is shown that, as the number of prototypes increases, the Euclidean distance metric yields better performance than the Mahalanobis distance. The experimental results of all recognition tasks demonstrate similar behaviour. It is probable that, as the number of prototypes increases, the Mahalanobis distance may require a large amount of training data to estimate the covariance matrices. Given a limited number of training samples, adding more prototypes asymptotically yields better results than introducing the weighting factors based on the covariance matrices.
- The $k$-means clustering algorithm, thus far, has been employed as the standard initialisation process in PBMEC. However, the inconsistency between the criteria used in the initialisation and recognition processes may unnecessarily affect the

recognition performance. With this in mind, we propose an alternative initialisation method, namely the $L_\nu$-$k$-means algorithm, which is the $L_\nu$-norm-based $k$-means algorithm. This proposed method results in an improvement in the recognition accuracy compared to the conventional $k$-means algorithm; therefore it is more suitable for PBMEC.

- The degree of smoothness in the decision boundaries also plays an important role in the MCE training process since it has a significant impact on the ability of PBMEC to generalise the unseen data. In the PBMEC training, the degree of smoothness can be controlled by the loss function tuning and the $\eta$ parameter in the computation of the misclassification measure. We utilise a sophisticated loss function proposed in [27], which does not require the tuning of loss parameters, because they are automatically determined from the training data. The $\eta$ parameter, on the other hand, allows us to control the degree of contributions among the competitive categories. Hence, its value should be carefully chosen. The results suggest that an approximation method, which sets $\eta$ to a very large value widely used in the literature, does not produce the optimal recognition performance.

- It is shown that our proposed techniques in PBMEC training demand significantly fewer parameters to achieve performance comparable to the $k$-NN classifier, based on similar feature extraction method. As a result, it is more suitable for deployment in the resource-constrained systems.

In comparison with the off-line scanning $n$-tuple and the convolutional networks classifiers, PBMEC yields less effective performance in the same recognition tasks. However, this result is not regarded as a fair and impartial comparison since the input of the other two classifiers is based on the bitmap images, which conveys whole information of the characters to the classifiers, whereas our PBMEC implementation relies on the compressed (only 24 dimensions) Discrete Cosine Transform (DCT) feature vectors. Obviously, much effort is required to select more powerful feature extraction methods. The applications of the discriminative feature extraction (DFE) to improve the feature extraction process are also described as an alternative method. PBMEC, together with our proposed techniques, has the potential to afford a compact classifier with high performance equivalent to other efficient classifiers.

# Chapter 9

# Conclusions

This chapter concludes the thesis with a summary of the present study (Section 9.1), followed by suggestions for future research in Section 9.2.

## 9.1 Research Summary

The most desirable property of handwriting recognition systems is their ability to cope with variations in writing style while distinguishing similar characters. In order to achieve this property, improvements can be made at any stage of the recognition process. This research particulary focuses on improving the classification stage, by introducing alternative training methods that enhance the discriminative abilities of the recognisers.

The conventional method for training the classifiers based on Maximum Likelihood (ML) estimation is presented in Chapter 2. The basis of ML training aims at building a model to represent the data of each class, by using the samples that belong to the corresponding class. It is regarded as a non-discriminative training method which needs to rely on the assumption that the correct functional form of the probability distribution is known. In contrast, discriminative training uses data from all competing classes when training the recogniser for each class. It attempts to directly determine the class decision boundaries through the optimisation of discriminative training criteria, without relying on the model correctness assumption. The chapter also outlines the rationale behind two popular discriminative training criteria: Maximum Mutual Information (MMI) and Minimum Classification Error (MCE). MMI training is intended to maximise the mutual information between the samples and their correct categories. On the other hand, MCE focuses on minimising the classification error during the training process.

One of the research contributions is that it demonstrates the effectiveness of discriminative training in handwriting recognition, which significantly improves the recognition performance of the non-discriminatively optimised systems. This important finding indicates that a gain in the recognition performance can be achieved without completely changing the classifier design, but rather through the use of the better training approach.

It is also essential as a foundation for future applications of discriminative training to other problems in handwriting recognition. In addition, the research covers the investigative and extensive studies of various aspects of discriminative training, which can be summarised in the following sections. The key contributions of this research can be found in Section 1.6.

### 9.1.1 Discriminative Training in HMMs

Hidden Markov Model (HMM) classifiers have been widely applied to handwriting recognition problems because of their ability to model continuous signals and capture statistical variations of natural handwriting. In these applications, HMMs are commonly trained by ML estimation via the powerful Baum-Welch optimisation algorithm, as discussed in Chapter 3. The concept of discriminative training in the HMMs is introduced as an alternative training method to ML training, with the emphasis on MMI training. The review of previous applications of MMI training in speech recognition has shown an improvement in the recognition performance when compared to ML training. However, there has been scant empirical evidence of its success in the handwriting recognition problem. The Baum-Welch algorithm is inapplicable to MMI training. The unavailability of the powerful optimisation algorithm may have prevented the extensive usage of MMI training in handwriting recognition.

For these reasons, Chapter 4 investigates various optimisation techniques for MMI training in the HMM-based handwriting recognition system, which include the EBW, Gradient Descent and QuickProp algorithms. We focus on the problem of on-line handwritten digit recognition. The secondary goal is to evaluate the performance of MMI training under a highly constrained environment, such as the system based on tied-mixture density HMMs (TDHMMs). The HMM classifier used in this study has a different structure from most applications of MMI training in the literature. The Gaussian parameters are shared among all states of the models, and optimising the Gaussian parameters in one model has an impact on those of other models. Among the three optimisation algorithms, the QuickProp algorithm proves to be the most efficient in the long run, although the Extended Baum-Welch gives the highest growth rate during the first few iterations.

The experiments show that MMI training is effective and improves the recognition performance when compared to ML training, despite being applied to such a highly tied system. In addition, the trade-off between using the tied-mixture system and the continuous density HMMs (CDHMMs), where there is no parameter tying involved, is revealed. The ML-trained TDHMM system yields higher recognition accuracy than that of the ML-trained CDHMM system. However, when MMI training is applied, the CDHMM system achieves a significantly higher gain in the recognition accuracy than the TDHMM one.

Finally, the experiments of investigating which HMM parameters yield maximum discriminative abilities are carried out. Although the highest performance is obtained when all HMM parameters are trained by MMI, it is found that optimising Gaussian parameters exhibits higher gain in the recognition accuracies than the mixture coefficient optimisation, in both the TDHMM and CDHMM contexts. This is rather counter-intuitive in the context of TDHMMs because the mixture coefficients are the only parameters which are not shared across the models and therefore expected to carry more discriminative abilities than the means and covariance matrices. Nevertheless, this finding suggests that, when designing the MMI training algorithms, much effort should be put into the optimisation of Gaussian parameters.

### 9.1.2 Thai Handwriting Recognition and MMI Training

The Thai characters pose a unique challenge for automatic handwriting recognition due to their similarity in shapes, with only slight variations in their small details, such as loops and curls. Chapter 5 gives an overview of the problem of Thai character recognition and a comprehensive review of previous research. The majority of previous studies rely on the presence of the character details as the important clues to identify the characters. However, these details are often attenuated in an unconstrained-writing style. Instead of searching for specially defined local structural features, we use the efficiently computed global features and a trainable classifier.

A high-performance system for off-line Thai handwriting recognition based on the HMMs and MMI training is introduced in Chapter 6. Much effort has been made to construct a database of Thai handwritten characters for this research, due to a lack of the standard database. To generate a sequence of feature vectors from the character image, the sliding-window technique is employed. Prior to performing MMI training, the feature extraction methods that yield reasonably good results on the ML-trained system need to be put in place. Extensive experiments are also carried out to find the most appropriate structure of the HMMs. The Gabor wavelet features yield the highest results among various investigated features, and are used throughout the experiments. We propose two different techniques: *block-based PCA* and *composite images*, which help improving the recognition performance of the system. The block-based PCA is an alternative way to apply PCA to the feature vectors that enhances the features extracted from vertical blocks in the frame image. On the other hand, the composite image is a concatenation of an original image with a 90-degree rotated image and a polar transformation of itself.

The MMI training experiments are carried out by using a variation of the EBW algorithm [10]. This algorithm is superior to the original EBW because it allows us to specify the smoothing parameters at the Gaussian level. The settings of these parameters are also more convenient than those in the QuickProp algorithm. The experiments demonstrate that the algorithm is effective and requires only a small number of MMI training iterations to achieve 100 % accuracy on the training set. However, only a slight increase in the recognition accuracy of the testing set is achieved, indicating that the system is not well generalised to the unseen data. We notice that the denominator likelihoods (the negative term in the MMI objective function) tend to be dominated by the correct character. To overcome this problem, we investigate the use of probability scaling in the MMI objective function which enables more characters to compete with the correct character. The compelling results show a significant improvement in the generalisation of the system. Finally, we introduce a more efficient way to compute the denominator likelihoods which takes into account only the $N$ most likely characters specified by the $N$-best lists. Not only does this technique reduce the computational time of MMI training, it also improves the recognition performance of the testing set. The best recognition result obtained from MMI training is 95.98% accuracy compared with 88.42% from ML training without the proposed methods, or equivalent to 65% relative error reduction. It is shown that the performance of our proposed system and the MMI training techniques for Thai handwriting recognition is slightly better than that achieved in the convolutional networks system.

### 9.1.3 Discriminative Training in Prototype-Based Classifiers

Apart from the application of discriminative training to the HMM-based recogniser, the present study shows that discriminative training is also effective when applied to the prototype-based handwriting recognition system. Because the prototype-based classifier is not explicitly probabilistic, the MCE criterion is deemed more applicable than the MMI criterion. Chapter 7 gives an overview of non-parametric classification techniques and presents the use of the MCE criterion in the prototype-based classifier, namely the Prototype-Based Minimum Error Classification (PBMEC), including a detail of its training process. PBMEC training is performed in two sequential stages: prototype initialisation and MCE training. Our extensive literature survey reveals that several important aspects of PBMEC training have never been addressed in the literature, such as the guideline for determining the values of smoothing parameters and the inconsistencies between the criteria used in the initialisation and recognition processes.

We take into account these aspects when applying PBMEC to the off-line handwriting recognition problem in Chapter 8. The PBMEC system is implemented and evaluated on the databases of isolated digits and English characters. The training process of PBMEC involves several settings of the smoothing parameters. For example, the contribution of each prototype to the PBMEC discriminant function is modulated by the parameter $\nu$ in the $L_\nu$-norm distance metric, and the degree of contribution among several competing categories is controlled by the parameter $\eta$ in the misclassification-measure computation. Most implementations of PBMEC in the literature rely on an approximation method which sets these parameters to $\infty$. Extensive experiments have revealed that such a method does not yield the optimal results since the recognition performance of PBMEC can be significantly improved when these parameters are carefully chosen. These parameters play an important role in controlling the degree of smoothness in the decision boundaries and have a significant impact on the ability of PBMEC to generalise the unseen data.

We propose a novel method for prototype initialisation in PBMEC to overcome the inconsistency between the criteria used in the initialisation and recognition processes. This method is based on the $L_\nu$-norm distance metric and demonstrates an improvement in the recognition accuracy when compared to the conventional $k$-means algorithm. It is shown that the PBMEC system requires a much smaller number of parameters than the $k$-NN classifier in order to achieve equivalent recognition performance. In addition, the potential application of discriminative training to the feature extraction process in the PBMEC context is explored and suggests that there is much room for improvement. In particular, emphasis should be placed on acquiring more powerful feature extraction methods.

## 9.2 Suggestions for Future Research

There is a wide range of potential research aspects to which the present study can be extended. These include the applications of discriminative training to different handwriting recognition problems and the improvements of existing techniques. Suggestions for future research areas are given in the following sections.

### 9.2.1 Discriminative Training and Cursive Handwriting Recognition

The results of discriminative training methods presented in this thesis are very promising. Even though the scope of the present study is limited to isolated character recognition, these methods can be potentially applied to cursive handwriting recognition problems. Given a vast wealth of literature regarding the HMM-based cursive handwriting recognition systems, it is anticipated that these methods may be of general use to improve the performance of those ML-trained systems.

In some application domains where a *lexicon* (a dictionary with restricted vocabulary) contains a very limited vocabulary (i.e., less than 200 words), such as the recognition of the currency amount written on bank cheques, each word in the lexicon can be modelled by an HMM. In this case, the problem is converted into isolated word recognition, and the techniques reported in this thesis can be directly applied. However, when the lexicon size becomes larger, isolated word recognition may not be appropriate, due to the lack of training data. HMM modelling should be performed at the character level in which two different approaches can be applied: *segmentation-based* and *segmentation-free*. The segmentation-based approach attempts to segment the handwritten word into characters. This approach converts the problem of cursive handwriting recognition into the isolated character recognition problem to which our techniques are applicable. Nevertheless, it requires a very powerful segmentation procedure in order to achieve good performance.

Since HMMs are capable of modelling continuous signals, the segmentation-free approach is deemed more appropriate. In this case, HMMs are modelled at the character basis, and the word model is constructed by concatenating corresponding character models, as indicated by its entry in the dictionary. Both ML and MMI training can be performed without knowing character boundaries. The implementation of MMI training is, however, more complicated because the denominator likelihoods need to be computed from the competing word models. $N$-best lists can be used to limit the computation to only the most likely competing word models. The use of *lattices* (see [10, 33]) is also promising as an alternative to the use of $N$-best lists. In addition, language models and prior probabilities can be employed to improve the performance of sentence recognition. These open up novel research avenues for the applications of MMI training to cursive handwriting recognition. Recent studies [4, 169] have provided baseline results for cursive handwriting recognition based on the ML-trained HMM systems with lexicons containing up to 50,000 words. Further evaluations of the performance of MMI training in these recognition tasks are required. It is also interesting to investigate the use of MCE training technique described in [43] on these tasks and compare the results with those of ML and MMI training.

An additional future research direction includes a derivation of novel discriminative training criteria that better suit to the cursive handwriting recognition problem. Moreover, PBMEC also has potential to be applied to cursive handwriting recognition, analogous to its applications to speech recognition, by incorporating a Dynamic Programming procedure and a multi-state architecture [17].

### 9.2.2 Thai Handwriting Recognition

The techniques introduced in this research can be directly applied to other HMM-based isolated handwriting recognition systems, such as the Sinhala handwriting recognition system presented in [125]. While the results presented in Chapter 6 are promising, they require handwritten texts to be segmented into individual Thai characters. When writing is done quickly and carelessly, the correct segmentation tends to be hardly achieved because contiguously written characters may connect to each other, yet without any change in each character's appearance. This is a difficult problem because touching Thai characters can occur at both the horizontal or vertical positions since some characters can be written above or below other characters. The method to tackle this problem is either to concentrate on improving the segmentation procedure or to utilise a segmentation-free approach. Analogous to cursive handwriting recognition, HMMs and discriminative training can be applied to a segmentation-free approach, as discussed in Section 9.2.1. However, this requires further explorations of techniques to model vertical touching characters by HMMs. It is also necessary to establish standard Thai handwriting databases with a large number of samples, both off-line and on-line, for future studies.

### 9.2.3 Improvements of Existing Techniques

In addition to the above suggestions, some aspects in the present study can be further improved. These require additional investigations summarised as follows:

- Further studies to evaluate the proposed techniques in Chapters 4, 6 and 8 on other standard databases, e.g. NIST databases [141, 142], are important to provide baseline results in order to compare with other recognition methods.
- Although various aspects of discriminative training have been studied in this thesis, we do not intend to directly compare the recognition performance among several discriminative training criteria. Future studies could take advantage of other discriminative training criteria proposed in the literature, such as Minimum Phone Error (MPE) [10] and Sigmoid-MMI (SMMI) [30], including MCE, and evaluate their recognition performance on the HMM-based handwriting recognition system.

- Apart from the optimisation algorithms for MMI training explored in this thesis, it is interesting to study other optimisation techniques, such as on-line learning methods [70], in greater depth.
- The proposed techniques based on block-based PCA and composite images should be further applied to other recognition tasks in order to investigate their general usage.
- Although the use of probability scaling in MMI training has significantly improved the generalisability of our recognition system, it is expected to yield less improvement when applied to cursive handwriting recognition, due to a higher variance in the data. Further investigations may be required to derive alternative methods that reduce the variance of the denominator statistics more efficiently.
- Much effort is required to obtain better feature extraction methods to be used in the PBMEC system. One possibility is to use the Gabor features described in Chapter 6. Also, the use of discriminative feature extraction (DFE) in PBMEC could be further improved by computing the transformed vector from the raw bitmap images. In this way, DFE could capture statistical variations among the character images, rather than the features which are already compressed (such as the DCT and the Gabor features). Another alternative is to use a non-linear feature transformation instead of the linear transformation.
- Further experiments to evaluate the performance of other prototype reduction techniques, such as reduced NN and LVQ, on the recognition problems in Chapter 8 would provide additional baseline results for the comparisons with the performance of PBMEC system.

# Appendix A

# Derivatives of the MMI Objective Function

This appendix presents the derivatives of the MMI objective function with respect to the mixture coefficients of the HMMs.

Consider a set of $R$ training samples $\mathbf{O} = \{\mathbf{O}_1^{T_1}, ..., \mathbf{O}_r^{T_r}, ..., \mathbf{O}_R^{T_R}\}$ and their correct category $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^r, ..., \mathcal{C}^R\}$, where $\mathcal{C}^r \in \{C_1, ..., C_M\}$, and $M$ is the total number of character classes. Each sample $\mathbf{O}_r^{T_r}$ is represented by a sequence of $T_r$ feature vectors, $\{\mathbf{o}_{r,1}, ..., \mathbf{o}_{r,t}, ...\mathbf{o}_{r,T_r}\}$. The model parameters of the HMM that represents character $C_i$ are denoted by $\lambda_i$. From Eq. 2.21, the MMI objective function can be rewritten here as:

$$f_{MMI} = \frac{1}{R} \sum_{r=1}^{R} \left( \log\, P(\mathbf{O}_r^{T_r} | \mathcal{C}^r) - \log \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r} | C_l) P(C_l) \right). \tag{A.1}$$

Note that the term $\frac{1}{R}$ in the above equation is normally omitted during the optimisation of $f_{MMI}$. Differentiating Eq. A.1 with respect to $c_{i,j,k}$, the $k$-th mixture coefficient of model $\lambda_i$ and state $s_j$, gives:

$$\frac{\partial f_{MMI}}{\partial c_{i,j,k}} = \sum_{r=1}^{R} \left( \frac{\partial \log\, P(\mathbf{O}_r^{T_r} | \mathcal{C}^r)}{\partial c_{i,j,k}} - \frac{\partial}{\partial c_{i,j,k}} \log \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r} | C_l) P(C_l) \right). \tag{A.2}$$

The computation of this derivative is done by considering each term in the brackets individually.

Consider the first term in Eq. A.2, the derivative of the log likelihood, $\log\, P(\mathbf{O}_r^{T_r} | \mathcal{C}^r)$, with respect to $c_{i,j,k}$. The calculation of likelihood probabilities can be approximated by using only the best state sequence obtained from the *Viterbi* path, as discussed in Section 4.2.2, instead of considering all possible state sequences. Given the state sequence $Q^{T_r} = \{q_1, ..., q_{T_r}\}$ obtained from the Viterbi path, where $q_t$ is the state at time $t$, this derivative can be computed from Eq. 3.6 as follows:

**Case 1:** when $\mathcal{C}^r = C_i$

$$\begin{aligned}\frac{\partial \log P(\mathbf{O}_r^{T_r}|\mathcal{C}^r)}{\partial c_{i,j,k}} &= \frac{\partial}{\partial c_{i,j,k}}\left(\sum_{t=2}^{T_r}\log a_{q_{t-1}q_t} + \sum_{t=1}^{T_r}\log b_{q_t}(\mathbf{o}_{r,t})\right)\\ &= \sum_{t=1}^{T_r}\frac{1}{b_{q_t}(\mathbf{o}_{r,t})}\frac{\partial b_{q_t}(\mathbf{o}_{r,t})}{\partial c_{i,j,k}}\end{aligned} \tag{A.3}$$

**Case 2:** when $\mathcal{C}^r \neq C_i$

$$\frac{\partial \log P(\mathbf{O}_r^{T_r}|\mathcal{C}^r)}{\partial c_{i,j,k}} = 0. \tag{A.4}$$

From Eq. 3.4, the output probability at state $s_j$ is calculated by:

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{K_j} c_{j,m}\mathcal{N}(\mathbf{o}_t, \boldsymbol{\mu}_{j,m}, \boldsymbol{\Sigma}_{j,m}), \tag{A.5}$$

where $K_j$ is the number of Gaussian mixtures in state $s_j$. Substituting the above equation into Eq. A.3 yields:

$$\frac{\partial \log P(\mathbf{O}_r^{T_r}|\mathcal{C}^r)}{\partial c_{i,j,k}} = \frac{1}{c_{i,j,k}}\sum_{t=1}^{T_r}\frac{c_{i,j,k}\mathcal{N}(\mathbf{o}_{r,t}, \boldsymbol{\mu}_{i,j,k}, \boldsymbol{\Sigma}_{i,j,k})}{\sum_{m=1}^{K_j} c_{i,j,m}\mathcal{N}(\mathbf{o}_{r,t}, \boldsymbol{\mu}_{i,j,m}, \boldsymbol{\Sigma}_{i,j,m})}, \tag{A.6}$$

where $q_t = s_j$ and $\mathcal{C}^r = C_i$.

By defining

$$\delta(r,i) = \begin{cases} 1 & ;\ \text{when } \mathcal{C}^r = C_i \\ 0 & ;\ \text{otherwise}\end{cases}, \tag{A.7}$$

and

$$\gamma^r_{i,j,k}(t) = \frac{c_{i,j,k}\mathcal{N}(\mathbf{o}_{r,t}, \boldsymbol{\mu}_{i,j,k}, \boldsymbol{\Sigma}_{i,j,k})}{\sum_{m=1}^{K_j} c_{i,j,m}\mathcal{N}(\mathbf{o}_{r,t}, \boldsymbol{\mu}_{i,j,m}, \boldsymbol{\Sigma}_{i,j,m})} \text{ when } q_t = s_j, \tag{A.8}$$

and $\gamma^r_{i,j,k}(t) = 0$ otherwise, we can reformulate Eq. A.3 and A.4 as:

$$\frac{\partial \log P(\mathbf{O}_r^{T_r}|\mathcal{C}^r)}{\partial c_{i,j,k}} = \frac{1}{c_{i,j,k}}\sum_{t=1}^{T_r}\gamma^r_{i,j,k}(t)\delta(r,i). \tag{A.9}$$

The term $\gamma^r_{i,j,k}(t)$ represents the probability of occupying the $k$-th mixture component in state $s_j$ of model $\lambda_i$. It is also known as an occupation count.

The next task is to compute the derivative in the second term of Eq. A.2. Since the prior probabilities $P(C_l)$ are constant, this derivative becomes:

$$\begin{aligned}\frac{\partial}{\partial c_{i,j,k}} \log \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)P(C_l) &= \frac{\partial}{\partial c_{i,j,k}} \log \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l) \\ &= \frac{1}{\sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)} \sum_{l=1}^{M} \frac{\partial P(\mathbf{O}_r^{T_r}|C_l)}{\partial c_{i,j,k}} \\ &= \frac{1}{\sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)} \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l) \frac{\partial \log\, P(\mathbf{O}_r^{T_r}|C_l)}{\partial c_{i,j,k}}.\end{aligned} \tag{A.10}$$

Given the fact that $\frac{\partial \log\, P(\mathbf{O}_r^{T_r}|C_l)}{\partial c_{i,j,k}} = 0$ when $l \neq i$, the above equation can be computed from Eq. A.9 as follows:

$$\frac{\partial}{\partial c_{i,j,k}} \log \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)P(C_l) = \frac{1}{c_{i,j,k}} \sum_{t=1}^{T_r} \frac{P(\mathbf{O}_r^{T_r}|C_i)}{\sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)} \gamma_{i,j,k}^{r}(t). \tag{A.11}$$

We define an *anti-occupation* count $\overline{\gamma}_{i,j,k}^{r}(t)$, which is obtained by passing the sample $\mathbf{O}_r^{T_r}$ into each of the competing models, including the correct one, as:

$$\overline{\gamma}_{i,j,k}^{r}(t) = \frac{P(\mathbf{O}_r^{T_r}|C_i)}{\sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)} \gamma_{i,j,k}^{r}(t). \tag{A.12}$$

Hence, the derivative in Eq. A.11 can be written as:

$$\frac{\partial}{\partial c_{i,j,k}} \log \sum_{l=1}^{M} P(\mathbf{O}_r^{T_r}|C_l)P(C_l) = \frac{1}{c_{i,j,k}} \sum_{t=1}^{T_r} \overline{\gamma}_{i,j,k}^{r}(t). \tag{A.13}$$

By substituting Eq. A.9 and A.13 into Eq. A.2, the final derivative of $f_{MMI}$ with respect to $c_{i,j,k}$ becomes:

$$\frac{\partial f_{MMI}}{\partial\, c_{i,j,k}} = \frac{1}{c_{i,j,k}} \sum_{r=1}^{R} \sum_{t=1}^{T_r} \left( \gamma_{i,j,k}^{r}(t)\delta(r,i) - \overline{\gamma}_{i,j,k}^{r}(t) \right). \tag{A.14}$$

Although the derivative presented in this appendix is performed in the context of the continuous density HMMs (CDHMMs), it is also applicable to the tied-mixture density HMM (TDHMM) system. In this latter case, the computation of the occupation counts is slightly different from Eq. A.8, as shown in Eq. 4.9.

# Bibliography

[1] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] G. Lorette. Handwriting Recognition or Reading? What is the Situation at the Dawn of the 3rd Millenium? *International Journal on Document Analysis and Recognition*, 2(1):2–12, 1999.

[3] H. Bunke. Recognition of Cursive Roman Handwriting – Past, Present and Future. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pages 448–459, Edinburgh, Scotland, 2003.

[4] A. Vinciarelli, S. Bengio and H. Bunke. Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):709–720, 2004.

[5] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, USA, 2002. MIT Press.

[6] Y. D. Rubinstein and T. Hastie. Discriminative vs Informative Learning. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 49–53, Newport Beach, California, USA, 1997.

[7] A. W. Senior. *Off-line Cursive Handwriting Recognition Using Recurrent Neural Networks.* PhD thesis, University of Cambridge, UK, September 1994.

[8] V. Sornlertlamvanich, T. Potipiti, C. Wutiwiwatchai and P. Mittrapiyanuruk. The State of the Art in Thai Language Processing. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, pages 597–598, Hong Kong, 2000.

[9] P. Bhattarakosol. IT Direction in Thailand: Cultivating an E-Society. *IT Professional*, 5(5):16–20, 2003.

[10] D. Povey. *Discriminative Training for Large Vocabulary Speech Recognition.* PhD thesis, University of Cambridge, UK, July 2004.

[11] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[12] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[13] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall International, New Jersey, 1984.

[14] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[15] J. L. Schafer. *Analysis of Incomplete Multivariate Data.* Chapman and Hall, New York, 1997.

[16] A. Nadas. A Decision Theoretic Formulation of a Training Problem in Speech Recognition, and a Comparison of Training by Conditional versus Unconditional Maximum Likelihood. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31(4):814–817, 1983.

[17] E. McDermott. *Discriminative Training for Speech Recognition.* PhD thesis, Waseda University, Japan, March 1997.

[18] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'86)*, pages 49–52, Tokyo, 1986.

[19] R. J. McEliece. *The Theory of Information and Coding.* Addison-Wesley, Massachusetts, 1977.

[20] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, New York, 1991.

[21] P. F. Brown. *The Acoustic-Modeling Problem in Automatic Speech Recognition.* PhD thesis, Carnegie Mellon University, Pittsburgh, May 1987.

[22] A. Nadas, D. Nahamoo, and M. Picheny. On a Model-Robust Training Method for Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(9):1432–1436, 1988.

[23] S. Amari. A Theory of Adaptive Pattern Classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307, 1967.

[24] S. Katagiri, C. H. Lee, and B.-H. Juang. New Discriminative Training Algorithms Based on the Generalized Descent Method. In *Proceedings of IEEE Worshop on Neural Networks for Signal Processing*, pages 299– 308, 1991.

[25] S. Katagiri, C. H. Lee, and B.-H. Juang. Discriminative Multilayer Feed-Forward Networks. In *Proceedings of IEEE Worshop on Neural Networks for Signal Processing*, pages 309–318, 1991.

[26] B.-H. Juang and S. Katagiri. Discriminative Learning for Minimum Error Classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3053, 1992.

[27] A. Biem. *Discriminative Feature Extraction applied to Speech Recognition.* PhD thesis, University of Paris 6, France, November 1997.

[28] S. Katagiri, B.-H. Juang and C. H. Lee. Pattern Recognition using a Family of Design Algorithms based upon the Generalized Probabilistic Descent Method. *Proceedings of the IEEE*, 86(11):2345–2373, 1998.

[29] S. Fahlman. An Empirical Study of Learning Speed in Back-Propagation Networks. Technical report, Department of Computer Science, Carnegie Mellon University, 1988.

[30] V. Valtchev. *Discriminative Methods in HMM-based Speech Recognition.* PhD thesis, University of Cambridge, UK, March 1995.

[31] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas and D. Nahamoo. Generalization of the Baum Algorithm to Rational Objective Functions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'89)*, pages 631–634, Glasgow, Scotland, 1989.

[32] Y. Normandin, R. Cardin, and R.D. Mori. High-Performance Connected Digit Recognition Using Maximum Mutual Information Estimation. *IEEE Transactions on Speech and Audio Processing*, 2(2):299–311, 1994.

[33] V. Valtchev, J. Odell, P. Woodland, and S. Young. MMIE Training of Large Vocabulary Recognition Systems. *Speech Communication*, 22(4):303–314, 1997.

[34] P. C. Woodland and D. Povey. Large Scale Discriminative Training of Hidden Markov Models for Speech Recognition. *Computer Speech and Language*, 16(1):25–48, 2002.

[35] K. Torkkola. Nonlinear Feature Transforms using Maximum Mutual Information. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'01)*, pages 2756–2761, Washington DC, USA, 2001.

[36] F. Beaufays, M. Weintraub, and Y. Konig. Discriminative Mixture Weight Estimation for Large Gaussian Mixture Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'99)*, pages 337–340, Phoenix, Arizona, USA, 1999.

[37] W. Chou, B.-H. Juang, and C. H. Lee. Segmental GPD Training of HMM Based Speech Recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'92)*, pages 473–476, San Francisco, California, USA, 1992.

[38] E. McDermott and S. Katagiri. Prototype-Based Discriminative Training for Various Speech Units. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'92)*, pages 417–420, San Francisco, California, USA, 1992.

[39] Y. Bengio, Y. LeCun, C. Nohl and C. Burges. LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. *Neural Computation*, 7(6):1289–1303, 1995.

[40] R. Zhang, X. Ding and J. Zhang. Offline Handwritten Character Recognition Based on Discriminative Training of Orthogonal Gaussian Mixture Model. In *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR'01)*, pages 221–225, Seattle, Washington, USA, 2001.

[41] W.-T. Chen and P. Gader. Word Level Discriminative Training For Handwritten Word Recognition. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition (IWFHR'00)*, pages 393–402, Amsterdam, The Netherlands, 2000.

[42] A. Biem. Minimum Classification Error Training of Hidden Markov Models for Handwriting Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'01)*, pages 1529–1532, Salt Lake City, Utah, USA, 2001.

[43] A. Biem. Minimum Classification Error Training for Online Handwritten Word Recognition. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, pages 61–66, Ontario, Canada, 2002.

[44] Q. Huo, Y. Ge and Z.-D. Feng. High Performance Chinese OCR Based on Gabor Features, Discriminative Feature Extraction and Model Training. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'01)*, pages 1517–1520, Salt Lake City, Utah, USA, 2001.

[45] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[46] F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, Massachusetts, USA, 1998.

[47] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, New Jersey, 1993.

[48] J. A. Bilmes. A Gentle Tutorial of the EM Algorithm and its Applications to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, International Computer Science Institute, Berkeley, California, 1998.

[49] R. Nag, K. H. Wong, and F. Fallside. Script Recognition using Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'86)*, pages 2071–2074, Tokyo, Japan, 1986.

[50] S. Bercu and G. Lorette. On-line Handwriting Word Recognition: An Approach based on Hidden Markov Models. In *Proceedings of the 3rd International Workshop on Frontiers in Handwriting Recognition (IWFHR'93)*, pages 385–390, Buffalo, New York, USA, 1993.

[51] K. S. Nathan, J. R. Bellegarda, D. Nahamoo and E. J. Bellegarda. On-line Handwriting Recognition Using Continuous Parameter Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'93)*, volume 5, pages 121–124, Minneapolis, Minnesota, USA, 1993.

[52] K. S. Nathan, H. S. M. Beigi, J. Subrahmonia, G. J. Clary and H. Maruyama. Real-time On-line Unconstrained Handwriting Recognition using Statistical Methods. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'95)*, volume 4, pages 2619–2623, Detroit, USA, 1995.

[53] A. Kosmala, J. Rottland and G. Rigoll. Improved On-Line Handwriting Recognition using Context Dependent Hidden Markov Models. In *Proceedings of the 4th International Conference on Document Analysis and Recognition (ICDAR'97)*, pages 641–644, Ulm, Germany, 1997.

[54] T. Starner, J. Makhoul, R. Schwartz and G. Chou. On-line Cursive Handwriting Recognition Using Speech Methods. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'94)*, volume 5, pages 125–128, Adelaide, Australia, 1994.

[55] J. Hu, M. K. Brown and W. Turin. Handwriting Recognition with Hidden Markov Models and Grammatical Constraints. In *Proceedings of the 4th International Workshop on Frontiers in Handwriting Recognition (IWFHR'94)*, pages 195–205, Taipei, Taiwan, 1994.

[56] J. Hu, M. K. Brown and W. Turin. HMM Based On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1039–1045, 1996.

[57] J. Hu, S. G. Lim and M. K. Brown. Writer Independent On-line Handwriting Recognition using an HMM Approach. *Pattern Recognition*, 33(1):133–147, 2000.

[58] C.-L. Liu, S. Jaeger and M. Nakagawa. Online Recognition of Chinese Characters: the State-of-the-Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):198–213, 2004.

[59] S. Jaeger, M. Nakagawa and C.-L. Liu. Comparing On-Line Recognition of Japanese and Western Script in Preparation for Recognizing Multi-Language Documents. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, pages 84–89, Ontario, Canada, 2002.

[60] B.-K. Sin and J. H. Kim. Ligature Modeling for Online Cursive Script Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):623–633, 1997.

[61] H. Bunke, M. Roth and E. G. Schukat-Talamazzini. Off-line Cursive Handwriting Recognition using Hidden Markov Models. *Pattern Recognition*, 28(9):1399–1413, 1995.

[62] D. Guillevic, C.Y. Suen. HMM Word Recognition Engine. In *Proceedings of the 4th International Conference on Document Analysis and Recognition (ICDAR'97)*, pages 544–547, Ulm, Germany, 1997.

[63] A. Vinciarelli and J. Luettin. Off-Line Cursive Script Recognition based on Continuous Density HMM. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition (IWFHR'00)*, pages 493–498, Amsterdam, The Netherlands, 2000.

[64] R. Plamondon and S. N. Srihari. On-line and Off-line Handwriting Recognition: a Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[65] A. Vinciarelli. A Survey on Off-line Cursive Word Recognition. *Pattern Recognition*, 35(7):1433–1446, 2002.

[66] A. Kundu and P. Bahl. Recognition of Handwritten Script: a Hidden Markov Model Based Approach. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'88)*, volume 2, pages 928–931, New York, USA, 1988.

[67] A. L. Koerich, R. Sabourin and C. Y. Suen. Large Vocabulary Off-line Handwriting Recognition: A Survey. *Pattern Analysis and Applications*, 6(2):97–121, 2003.

[68] A. L. Koerich, R. Sabourin and C. Y. Suen. Lexicon-driven HMM Decoding for Large Vocabulary Handwriting Recognition with Multiple Character Models. *International Journal on Document Analysis and Recognition*, 6(2):126–144, 2003.

[69] B. Merialdo. Phonetic Recognition Using Hidden Markov Models and Maximum Mutual Information Training. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'88)*, pages 111–114, New York, USA, 1988.

[70] S. Kapadia. *Discriminative Training of Hidden Markov Models.* PhD thesis, University of Cambridge, UK, March 1998.

[71] Y. Normandin and S. D. Morgera. An Improved MMIE Training Algorithm for Speaker-independent, Small Vocabulary, Continuous Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'91)*, volume 1, pages 537–540, Toronto, Canada, 1991.

[72] Y. Normandin. *Hidden Markov Models, Maximum Mutual Information Estimation and the Speech Recognition Problem.* PhD thesis, McGill University, Canada, 1991.

[73] R. Schluter. *Investigations on Discriminative Training Criteria.* PhD thesis, RWTH Aachen, Germany, September 2000.

[74] R. Schluter, W. Macherey, S. Kanthak, H. Ney and L. Welling. Comparison of Optimization Methods for Discriminative Training Criteria. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH'97)*, volume 1, pages 15–18, Rhodes, Greece, 1997.

[75] J. Zheng, J. Butzberger, H. Franco and A. Stolcke. Improved Maximum Mutual Information Estimation Training of Continuous Density HMMs. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH'01)*, volume 2, pages 679–682, Aalborg, Denmark, 2001.

[76] Y.-L. Chow. Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition using the N-best Algorithm. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'90)*, volume 2, pages 701–704, New Mexico, USA, 1990.

[77] H. S. M. Beigi, K. Nathan, G. J. Clary and J. Subrahmonia. Size Normalization in On-line Unconstrained Handwriting Recognition. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'94)*, volume 1, pages 169–173, Austin, USA, 1994.

[78] H. S. M. Beigi, K. Nathan and J. Subrahmonia. On-Line Unconstrained Handwriting Recognition Based on Probabilistic Techniques. In *Proceedings of the 4th Iranian Conference on Electrical Engineering (ICEE'95)*, Tehran, Iran, 1995.

[79] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo and K. S. Nathan. A Fast Statistical Mixture Algorithm for On-line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1227–1233, 1994.

[80] J. Subrahmonia, K. Nathan and M. P. Perrone. Writer Dependent Recognition of On-line Unconstrained Handwriting. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'96)*, volume 6, pages 3478–3481, Atlanta, Georgia, USA, 1996.

[81] A. Senior, J. Subrahmonia and K. Nathan. Duration Modeling Results for an On-line Handwriting Recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'96)*, volume 6, pages 3482–3485, Atlanta, Georgia, USA, 1996.

[82] K. S. Nathan, J. Subrahmonia and M. P. Perrone. Parameter Tying in Writer-Dependent Recognition of On-line Handwriting. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96)*, volume 3, pages 28–32, Vienna, Austria, 1996.

[83] J. R. Bellegarda and D. Nahamoo. Tied Mixture Continuous Parameter Modeling for Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(12):2033–2045, 1990.

[84] S. Young. A Review of Large Vocabulary Continuous Speech Recognition. *IEEE Signal Processing Magazine*, 13(5):45–57, 1996.

[85] R. Nopsuwanchai and A. Biem. Discriminative Training of Tied Mixture Density HMMs for Online Handwritten Digit Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'03)*, volume 2, pages 817–820, Hong Kong, China, 2003.

[86] B. Comrie (ed.). *The World's Major Languages.* Oxford University Press, New York, 1990.

[87] N. Ronnakiat. Thai Writing System: History. *The Encyclopedia of Language and Linguistics*, 9:4601–4602, 1994.

[88] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley, Reading, MA, USA, 1992.

[89] S. Duangphasuk and N. Premchaiswadi. Document Image Skew Detection. In *Proceedings of the 3rd National Computer Science and Engineering Conference (NCSEC'99)*, pages 146–151, Bangkok, Thailand, 1999 (in Thai).

[90] J. R. Parker. *Algorithms for Image Processing and Computer Vision.* Wiley, New York, USA, 1996.

[91] C. Santinanalert. *Design and Development of Thai-OCR Program.* Master's thesis, Chulalongkorn University, Bangkok, Thailand, 1999 (in Thai).

[92] S. Airphaiboon and S. Kondo. Recognition of Handprinted Thai Characters Using Loop Structures. *IEICE Transactions on Information and Systems*, E79-D(9):1296–1303, 1996.

[93] P. Veerathanabutr and K. Homma. The Off-Line Thai Handwritten Character Recognition. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 124–135, Chiang Mai, Thailand, 2000.

[94] P. Limmaneewichid and N. Premchaiswadi. Repairing Broken Thai Printed Characters Using Feature Extraction. In *Proceedings of the 3rd National Computer Science and Engineering Conference (NCSEC'99)*, pages 152–157, Bangkok, Thailand, 1999 (in Thai).

[95] C. Kimpan, A. Itoh and K. Kawanishi. Recognition of Printed Thai Characters Using a Matching Method. *IEE Proceedings, Part E: Computers and Digital Techniques*, 130(6):183–188, 1983.

[96] U. Marang, P. Phokharatkul and C. Kimpan. Recognition of Printed Thai Characters Using Boundary Normalization and Fuzzy Neural Networks. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 154–164, Chiang Mai, Thailand, 2000.

[97] P. Phokharatkul and C. Kimpan. Handwritten Thai Character Recognition Using Fourier Descriptors and Genetic Neural Networks. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 108–123, Chiang Mai, Thailand, 2000.

[98] N. Premchaiswadi, W. Premchaiswadi and S. Narita. Segmentation of Horizontal and Vertical Touching Thai Characters. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E83-A(6):987–995, 2000.

[99] S. Chaiyakhet and K. Chamnongthai. Off-line Handwritten Thai Character Segmentation. In *Proceedings of the 3rd National Computer Science and Engineering Conference (NCSEC'99)*, pages 250–254, Bangkok, Thailand, 1999 (in Thai).

[100] V. Premratanachai, W. Premchaiswadi and N. Premchaiswadi. Crossing Thai Characters Segmentation. In *Proceedings of the 3rd National Computer Science and Engineering Conference (NCSEC'99)*, pages 240–249, Bangkok, Thailand, 1999 (in Thai).

[101] C. Kimpan, A. Itoh and K. Kawanishi. Fine Classification of Printed Thai Character Recognition Using the Karhunen-Loeve Expansion. *IEE Proceedings, Part E: Computers and Digital Techniques*, 134(5):257–264, 1987.

[102] S. Srisuk. Thai Printed Character Recognition Using the Hausdorff Distance. In *Proceedings of the 3rd National Computer Science and Engineering Conference (NCSEC'99)*, pages 233–239, Thailand, 1999.

[103] P. Hiranvanichakorn, T. Agui and M. Nakajima. A Recognition Method of Thai Characters. *Journal of the Institute of Electronics and Communication Engineers of Japan (IECE)*, E65(12):737–744, 1982.

[104] P. Hiranvanichakorn, T. Agui and M. Nakajima. Recognition Method of Thai Characters by Using Local Features. *Journal of the Institute of Electronics and Communication Engineers of Japan (IECE)*, E67(8):425–432, 1984.

[105] C. Kimpan. Printed Thai Character Recognition Using Topological Properties Method. *International Journal of Electronics*, 60(3):303–329, 1986.

[106] B. Kijsirikul and S. Sinthupinyo. Approximate ILP Rules by Backpropagation Neural Network: A Result on Thai Character Recognition. In *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 162–173, Bled, Slovenia, 1999.

[107] S. Sinthupinyo and B. Kijsirikul. Approximation of First-Order Rules by the Backpropagation Neural Network. In *Proceedings of the 4th National Computer Science and Engineering Conference (NCSEC2000)*, Bangkok, Thailand, 2000.

[108] A. Kawtrakul and P. Waewsawangwong. Multi-Feature Extraction for Printed Thai Character Recognition. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 301–308, Chiang Mai, Thailand, 2000.

[109] C. Y. Suen, K. Kim, Q. Xu, J. Kim and L. Lam. Handwriting Recognition - The Last Frontiers. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR'00)*, volume 4, pages 4001–4010, Barcelona, Spain, 2000.

[110] P. Hiranvanichakorn, T. Agui and M. Nakajima. A Recognition Method of Handprinted Thai Characters by Local Features. *Journal of the Institute of Electronics and Communication Engineers of Japan (IECE)*, E68(2):83–90, 1985.

[111] I. Methasate, S. Jitapunkul, K. Kiratiratanaphrug and W. Unsiam. Fuzzy Feature Extraction for Thai Handwritten Character Recognition. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 136–141, Chiang Mai, Thailand, 2000.

[112] S. Arunrungrusmi and K. Chamnongthai. Adaptive Blocks for Skeleton Segmentation in Handwritten Thai Character. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 309–316, Chiang Mai, Thailand, 2000.

[113] W. Chatwiriya. *Off-line Thai Handwriting Recognition in Legal Amount.* PhD thesis, West Virginia University, West Virginia, USA, January 2002.

[114] P. Hiranvanichakorn, T. Agui and M. Nakajima. An On-line Recognition Method of Thai Characters. *Journal of the Institute of Electronics and Communication Engineers of Japan (IECE)*, E68(9):594–601, 1985.

[115] P. Kortungsap, P. Lekhachaivorakul and S. Madarasmi. On-line Handwriting Recognition System for the Thai, English, Numeral and Symbol Characters. In *Proceedings of the 3rd National Computer Science and Engineering Conference (NCSEC'99)*, pages 255–259, Bangkok, Thailand, 1999.

[116] S. Madarasmi and P. Lekhachaiworakul. Customizable On-line Thai/English Handwriting Recognition System. In *Proceedings of the 4th Symposium on Natural Language Processing (SNLP2000)*, pages 142–153, Chiang Mai, Thailand, 2000.

[117] I. Methasate and S. Sae-Tang. On-line Thai Handwriting Character Recognition Using Stroke Segmentation with HMM. In *Proceedings of the 20th IASTED International Conference on Applied Informatics - Artificial Intelligence and Applications (AI2002)*, pages 59–62, Innsbruck, Austria, 2002.

[118] S. Sae-Tang and I. Methaste. Thai Online Handwritten Character Recognition Using Windowing Backpropagation Neural Networks. In *Proceedings of the 20th IASTED International Conference on Modelling, Identification, and Control (MIC2002)*, pages 337–340, Innsbruck, Austria, 2002.

[119] C. Pornpanomchai, D. N. Batanov and N. Dimmitt. Recognizing Thai Handwritten Characters and Words for Human-Computer Interaction. *International Journal of Human Computer Studies*, 55(3):259–279, 2001.

[120] R. Jain, R. Kasturi and B. G. Schunck. *Machine Vision.* McGraw-Hill, New York, USA, 1995.

[121] M. S. M. Khorsheed. *Automatic Recognition of Words in Arabic Manuscripts.* PhD thesis, University of Cambridge, UK, June 2000.

[122] 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 Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[123] A. Kornai. An Experimental HMM-Based Postal OCR System. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'97)*, volume 4, pages 3177–3180, Los Alamitos, California, USA, 1997.

[124] W. Wang, A. Brakensiek, A. Kosmala and G. Rigoll. HMM Based High Accuracy Off-line Cursive Handwriting Recognition by a Baseline Detection Error Tolerant Feature Extraction Approach. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition (IWFHR'00)*, pages 209–218, Amsterdam, The Netherlands, 2000.

[125] S. Hewavitharana, H. C. Fernando and N. D. Kodikara. Off-Line Sinhala Handwriting Recognition Using Hidden Markov Models. In *Proceedings of the 3rd Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'02)*, Ahmedabad, India, 2002.

[126] N. Arica and F. T. Yarman-Vural. Optical Character Recognition for Cursive Handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):801–813, 2002.

[127] R. Nopsuwanchai and W. F. Clocksin. Hidden Markov Models for Off-line Thai Handwriting Recognition. In *Proceedings of the 11th International Conference on Artificial Intelligence Applications (ICAIA'03)*, pages 180–189, Cairo, Egypt, 2003.

[128] Ø. D. Trier, A. K. Jain and T. Taxt. Feature Extraction Methods for Character Recognition - A Survey. *Pattern Recognition*, 29(4):641–662, 1996.

[129] M.-K. Hu. Visual Pattern Recognition by Moment Invariants. *IEEE Transactions on Information Theory*, 8(2):179–187, 1962.

[130] S. X. Liao. *Image Analysis by Moments.* PhD thesis, University of Manitoba, Winnipeg, Manitoba, Canada, 1993.

[131] J. Flusser and T. Suk. Affine Moment Invariants: A New Tool for Character Recognition. *Pattern Recognition Letters*, 15(4):433–436, 1994.

[132] J. G. Daugman. Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-dimensional Visual Cortical Filters. *Journal of the Optical Society of America A: Optics, Image Science & Vision*, 2(7):1160–1169, 1985.

[133] J. G. Daugman. Complete Discrete 2D Gabor Transform by Neural Networks for Image Analysis and Compression. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(7):1169–1179, 1988.

[134] M. Lades, J. C. Vorbrüggen, J. Buhmann, J. Lange, C.v.d. Malsburg, R. P. Würtz and W. Konen. Distortion Invariant Object Recognition in the Dynamic Link Architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.

[135] Y. Hamamoto, S. Uchimura, M. Watanabe, T. Yasuda, Y. Mitani and S. Tomita. A Gabor Filter-based Method for Recognizing Handwritten Numerals. *Pattern Recognition*, 31(4):395–400, 1998.

[136] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev and P. Woodland. *The HTK Book (for HTK Version 3.2).* Cambridge University Engineering Department, Cambridge, UK, 2002.

[137] R. O. Duda, P. E. Hart and D. G. Stork. *Pattern Classification.* Wiley, New York, 2001.

[138] A. Vinciarelli and S. Bengio. Offline Cursive Word Recognition using Continuous Density Hidden Markov Models Trained with PCA or ICA Features. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, volume 3, pages 81–84, Québec City, Canada, 2002.

[139] R. Nopsuwanchai and D. Povey. Discriminative Training for HMM-Based Offline Handwritten Character Recognition. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pages 114–118, Edinburgh, Scotland, 2003.

[140] M. Tistarelli and G. Sandini. On the Advantages of Polar and Log-polar Mapping for Direct Estimation of Time-to-impact from Optical Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):401–410, 1993.

[141] R. A. Wilkinson, J. Geist, S. Janet, P. J. Grother, C. J. C. Burges, R. Creecy, B. Hammond, J. J. Hull, N. J. Larsen, T. P. Vogl and C. L. Wilson. *The First Census Optical Character Recognition Systems Conference*. Technical report, National Institute of Standards and Technology, Gaithersburg, Maryland, USA, September 1992.

[142] J. Geist, R. A. Wilkinson, S. Janet, P. J. Grother, B. Hammond, N. W. Larsen, R. M. Klear, C. J. C. Burges, R. Creecy, J. J. Hull, T. P. Vogl and C. L. Wilson. *The Second Census Optical Character Recongition Systems Conference*. Technical report, National Institute of Standards and Technology, Gaithersburg, Maryland, USA, June 1994.

[143] T. M. Cover and P. E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, 1967.

[144] F. Ricci and P. Avesani. Data Compression and Local Metrics for Nearest Neighbor Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):380–384, 1999.

[145] T. Hong, S. W. Lam, J. J. Hull and S. N. Srihari. The Design of a Nearest-Neighbor Classifier and Its Use for Japanese Character Recognition. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR'95)*, pages 270–273, Montreal, Canada, 1995.

[146] C.-L. Liu and M. Nakagawa. Evaluation of Prototype Learning Algorithms for NearestNeighbor Classifier in Application to Handwritten Character Recognition. *Pattern Recognition*, 34(3):601–615, 2001.

[147] G. W. Gates. The Reduced Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 18(5):431–433, 1972.

[148] P. E. Hart. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.

[149] P. A. Devijver and J. Kittler. On the Edited Nearest Neighbor Rule. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 72–80, Miami, Florida, USA, 1980.

[150] D. B. Skalak. *Prototype Selection for Composite Nearest Neighbor Classifiers*. PhD thesis, University of Massachusetts Amherst, Massachusetts, May 1997.

[151] T. Kohonen, G. Barna and R. Chrisley. Statistical Pattern Recognition with Neural Networks: Benchmarking Studies. In *Proceedings of IEEE Conference on Neural Networks*, volume 1, pages 61–68, San Diego, California, USA, 1988.

[152] T. Kohonen. The Self-Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[153] E. McDermott. Discriminative Prototype-Based Methods for Speech Recognition. In *Handbook of Neural Networks for Speech Processing*, Editted by S. Katagiri, Artech House, 2000.

[154] A. Biem and S. Katagiri. Cepstrum-Based Filter-Bank Design Using Discriminative Feature Extraction Training at Various Levels. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 1503–1506, Munich, Germany, 1997.

[155] D. Luenberger. *Optimization by Vector Space Methods.* John Wiley & Sons, New York, 1998.

[156] A. K. Jain, R. P. W. Duin and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[157] T. Komori and S. Katagiri. GPD Training of Dynamic Programming-based Speech Recognizers. *Journal of the Acoustical Society of Japan (ASJ)*, 13(6):341–349, 1992.

[158] P. C. Chang and B.-H. Juang. Discriminative Training of Dynamic Programming based Speech Recognizers. *IEEE Transactions on Speech and Audio Processing*, 1(2):135–143, 1993.

[159] E. McDermott and S. Katagiri. Prototype-Based MCE/GPD Training for Word Spotting and Connected Word Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'93)*, pages 291–294, Minneapolis, Minnesota, USA, 1993.

[160] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman and S. Janet. UNIPEN Project of On-line Data Exchange and Recognizer Benchmarks. In *Proceedings of the 12th International Conference on Pattern Recognition (ICPR'94)*, pages 72–80, Jerusalem, Israel, 1994.

[161] E. H. Ratzlaff. Methods, Report and Survey for the Comparison of Diverse Isolated Character Recognition Results on the UNIPEN Database. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pages 623–628, Edinburgh, Scotland, 2003.

[162] E. H. Ratzlaff. A Scanning n-tuple Classifier for Online Recognition of Handwritten Digits. In *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR'01)*, pages 18–22, Seattle, Washington, USA, 2001.

[163] E. Feig and S. Winograd. Fast Algorithms for the Discrete Cosine Transform. *IEEE Transactions on Signal Processing*, 40(9):2174–2193, 1972.

[164] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press, Boston, 1990.

[165] R. Nopsuwanchai and A. Biem. Prototype-Based Minimum Error Classifier for Handwritten Digits Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'04)*, volume 5, pages 845–848, Montreal, Canada, 2004.

[166] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, Massachusetts, USA, 1981.

[167] A. Biem and S. Katagiri. Feature Extraction Based on Minimum Classification Error/Generalised Probabilistic Descent Method. In *Proceedings of the IEEE In-*

*ternational Conference on Acoustics, Speech and Signal Processing (ICASSP'93)*, pages 275–278, Minneapolis, Minnesota, USA, 1993.

[168] A. Biem and S. Katagiri. Filter Bank Design Based on Discriminative Feature Extraction. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'94)*, pages 485–488, Adelaide, Australia, 1994.

[169] A. Vinciarelli. *Offline Cursive Handwriting: From Word to Text Recognition.* PhD thesis, IDIAP, Martigny, Switzerland, April 2003.