UNIVERSITY OF CALIFORNIA, SAN DIEGO

Online Learning of Large Margin Hidden Markov Models for Automatic Speech Recognition

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Chih-Chieh Cheng

Committee in charge:

Professor Lawrence K. Saul, Chair
Professor Sanjoy Dasgupta
Professor Charles Elkan
Professor Gert Lanckriet
Professor Bhaskar Rao
Professor Fei Sha

2011

The dissertation of Chih-Chieh Cheng is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2011

DEDICATION

To my parents, who never have a chance to attend college but gave their kids every opportunity to pursue their dreams.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First I would like to thank my advisor, Prof. Lawrence Saul, for his support and advising through the many years, and for inspiring me to work so many long nights before paper deadlines. I also appreciate his help in drafting this thesis. It is my fortune to have such a great advisor and a friend for life.

I would also like to acknowledge my committee members, Prof. Charles Elkan, Prof. Sanjoy Dasgupta, Prof. Gert Lanckriet, Prof. Bhaskar Rao, and Prof. Fei Sha. Thanks for their valuable comments and suggestions to this thesis.

I would like to give special thanks to Prof. Fei Sha and Dr. Brian Kingsbury for their support throughout my PhD program. I thank Prof. Fei Sha for being like an academic big brother and for sharing all his research experience. I thank Brian for his valuable guidance during my internship in IBM and his generous suggestions in research.

Thanks also to my dear labmates: Diane Hu, Youngmin Cho, Justin Ma, Shibin Parameswaran, and Laurens van der Maaten. Thank you, Diane, for entertaining me through the difficult periods of graduate school, and thank you for screaming with me, gossiping with me, and sharing gorgeous shopping sites with me. I thank Youngmin for his hard-working spirit and the sound of his constant typing, which drove me to concentrate on my own work. I thank Shibin for sharing his awesome Taekwondo as exercise in the office. (Whom were we defending?) I thank Justin for bugging us with stuffy animals and fail stamps. I thank Laurens for his great European humor. I owe many thanks to all the people whom I have worked with, in the lab, in classes, and in research. I also want to thank all my dear friends, wherever they are.

Thanks to Fr. Jaime Valenciano S.J. for his consistent guidance across the Pacific Ocean throughout these years. Thanks to my church group for their great support. Your prayers give me so much strength and warmth when I am away from home.

Finally, I would like to give honor to my parents. Without them, I would not be where I am today. They did not have too many opportunities when they were young, but they gave me everything I needed – freedom, support, and trust – to support the pursuit of my dreams. They did a greater job than getting a PhD degree. Last but not least, I want to thank Marcus for being the great partner of my life.

Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 6, in part, is a reprint of the material as it appear in Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML-09) 2009. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, ACM, 2009. Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of these papers.

Chapter 7, in part, is a reprint of the material as it appear in Proceedings of the Tenth Annual Conference of the International Speech Communication Association (Interspeech-09) 2009. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, International Speech Communication Association (ISCA), 2009. Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of these papers.

Chapter 8, in part, is a reprint of the material as it appear in Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-09) 2009. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE, 2009. Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of these papers.

VITA

2002	B. S. in Computer Science, National Tsing Hua University, Taiwan
2004	M. S. in Computer Science, National Tsing Hua University, Taiwan
2004-2006	Software Engineer in Computer and Communication Lab, Industrial Technology and Research Institute, Hsinchu, Taiwan
2011	Ph. D. in Computer Science, University of California, San Diego, USA

PUBLICATIONS

Joseph Keshet, Chih-Chieh Cheng, Mark Stoehr, David McAllester, Lawrence K. Saul (2011). **Direct Error Rate Minimization of Hidden Markov Models.** To appear in *Proceedings of INTERSPEECH-2011.*

Chih-Chieh Cheng and Brian Kingsbury (2011). **Arccosine Kernels: Acoustic Modeling with Infinite Neural Networks.** To appear in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-11).*

Chih-Chieh Cheng, Fei Sha, and Lawrence K. Saul (2010). **Online Learning and Acoustic Feature Adaptation in Large Margin Hidden Markov Models.** In *IEEE Journal of Selected Topics in Signal Processing* 4(6): 926-942.

Chih-Chieh Cheng, Fei Sha, and Lawrence K. Saul (2009). **Large Margin Feature Adaptation for Automatic Speech Recognition.** In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-09)*, pages 87-92. Merano, Italy.

Chih-Chieh Cheng, Fei Sha, and Lawrence K. Saul (2009). **A fast online algorithm for large margin training of continuous-density hidden Markov models.** In *Proceedings of the Tenth Annual Conference of the International Speech Communication Association (Interspeech-09)*, pages 668-671. Brighton, UK.

Chih-Chieh Cheng, Fei Sha, and Lawrence K. Saul (2009). **Matrix updates for perceptron training of continuous-density hidden Markov models.** In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML-09)*, pages 153-160. Montreal, Canada.

Chih-Chieh Cheng, Diane J. Hu, and Lawrence K. Saul (2008). **Nonnegative Matrix Factorization for Real Time Musical Analysis and Sight-Reading Evaluation.** In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-08)*, pages 2017-2020. Las Vegas, NV.

J. Matteson, C. K. Kha, D. J. Hu, C.-C. Cheng, L. K. Saul, and G. R. Sadler (2008). **Campus community partnerships with people who are deaf or hard of hearing.** *Assistive Technology Outcomes and Benefits* 5(1): 29-44.

Chih-Chieh Cheng and Chiou-Ting Hsu (2006). **Fusion of Audio and Motion Information on HMM-Based Highlight Extraction for Baseball Games.** *IEEE Trans. Multimedia*, 8(3): 585-599.

Chih-Chieh Cheng and Chiou-Ting Hsu (2002). **Content-Based Audio Classification with Generalized Ellipsoid Distance.** In *Proceedings of the Third IEEE Pacific-Rim Conference on Multimedia (PCM 2002)*, pages: 328-335. Hsinchu, Taiwan.

ABSTRACT OF THE DISSERTATION

Online Learning of Large Margin Hidden Markov Models for Automatic Speech Recognition

by

Chih-Chieh Cheng

Doctor of Philosophy in Computer Science

University of California, San Diego, 2011

Professor Lawrence K. Saul, Chair

Over the last two decades, large margin methods have yielded excellent performance on many tasks. The theoretical properties of large margin methods have been intensively studied and are especially well-established for support vector machines (SVMs). However, the scalability of large margin methods remains an issue due to the amount of computation they require. This is especially true for applications involving sequential data.

In this thesis we are motivated by the problem of automatic speech recognition (ASR) whose large-scale applications involve training and testing on extremely large data sets. The acoustic models used in ASR are based on continuous-density hidden Markov models (CD-HMMs). Researchers in ASR have focused on discriminative

training of HMMs, which leads to models with significantly lower error rates. More recently, building on the successes of SVMs and various extensions thereof in the machine learning community, a number of researchers in ASR have also explored large margin methods for discriminative training of HMMs.

This dissertation aims to apply various large margin methods developed in the machine learning community to the challenging large-scale problems that arise in ASR. Specifically, we explore the use of sequential, mistake-driven updates for online learning and acoustic feature adaptation in large margin HMMs. The updates are applied to the parameters of acoustic models after the decoding of individual training utterances. For large margin training, the updates attempt to separate the log-likelihoods of correct and incorrect transcriptions by an amount proportional to their Hamming distance. For acoustic feature adaptation, the updates attempt to improve recognition by linearly transforming the features computed by the front end. We evaluate acoustic models trained in this way on the TIMIT speech database. We find that online updates for large margin training not only converge faster than analogous batch optimizations, but also yield lower phone error rates than approaches that do not attempt to enforce a large margin.

We conclude this thesis with a discussion of future research directions, highlighting in particular the challenges of scaling our approach to the most difficult problems in large-vocabulary continuous speech recognition.

Chapter 1

Introduction

This thesis aims to apply large margin methods developed in the machine learning community to the challenging large-scale problems that arise in automatic speech recognition (ASR). Over the last two decades, large margin methods have yielded excellent performance on many tasks. However, the scalability of large margin methods remains an issue due to the amount of computation they require. This is especially true for applications involving sequential data. In this thesis, the demands of large-scale ASR motivate us to investigate online methods for large margin classification of sequential data.

1.1 Motivation

Most existing systems for ASR are based on continuous-density hidden Markov models (CD-HMMs), whose parameters must be estimated from large training corpora of transcribed speech [HAH01]. The simplest approach to this problem is maximum likelihood (ML) estimation, which attempts to maximize the joint likelihood of the training data. However, ML estimation has well-known limitations for ASR. At best, CD-HMMs provide only an approximate model of the tremendous variability observed in real speech. When such models are estimated from training data, improvements in their joint likelihoods do not always translate into fewer recognition errors. This realization has led researchers to develop other objective functions for parameter estimation that more closely track the error rate (however it is measured).

A great deal of research in ASR has focused on discriminative training of HMMs [BBdSM86, Nad83, JK92]. Perhaps the most popular framework for discriminative training is maximum mutual information (MMI) estimation. In this framework, model parameters are estimated to maximize the mutual information between the desired recognizer output and the acoustic features computed by the front end. More recently, building on the successes of support vector machines [CV95, Vap98] and various extensions thereof [THJA04, TGK04] in the machine learning community, a number of researchers in ASR have also explored large margin methods for discriminative training of HMMs [JLL06a, LYL07, YDHA07, SS09].

In ASR, a major challenge of discriminative training arises from the combinatorially large number of possible phonetic transcriptions per speech utterance. To succeed, discriminative methods must separate the likelihood of the correct decoding from all incorrect hypotheses. The need to consider incorrect hypotheses makes discriminative training much more computationally intensive than ML estimation. Large corpora are typically managed by parallelizing batch computations of parameter updates across many different nodes, then combining the individual results to average over training utterances. For large-vocabulary ASR, discriminative training can also be accelerated by using lattices to provide a compact representation of alternative hypotheses [WP00]. Nevertheless, the scaling of discriminative methods to large-scale problems remains an important area for ongoing research.

A similar problem of scaling confronts researchers in machine learning, whose algorithms must deal with data sets of ever-increasing size. The demands of large-scale applications have led to a resurgence of interest in *online* learning algorithms [BL04, BB08]. These algorithms update model parameters after the presentation of each labeled example, thus eliminating the need to store or manipulate the entire data set in memory. Not only are these online algorithms simpler to implement and more feasible for large-scale learning, but in many cases they converge more quickly and perform better than their batch counterparts.

1.2 Overview

Motivated by the potential of this approach for ASR, in this thesis we investigate an online algorithm for discriminative training of HMMs. The algorithm optimizes the parameters of acoustic models in an incremental fashion, updating them after the decoding of each training utterance. The first main contribution of this thesis is to propose a particular reparameterization of acoustic models that lends itself very well to this type of training. We present experimental results for acoustic models trained in this way on the TIMIT speech corpus [LKS86]. The TIMIT corpus is a small-scale but still widely used benchmark [GMAP05, PPK07, SSL07, DYA06] for evaluating new approaches to hidden Markov modeling in ASR. We systematically compare the effects of different parameterizations, initializations, and averaging schemes on convergence rates and phone recognition accuracies. Our results illustrate a set of best practices for online, discriminative training that yield the most consistently significant and rapid reductions in phone recognition error rates [CSS09c].

The second main contribution of this thesis is to investigate online updates for large margin training of HMMs [JLL06a, SS07b, SS07a, LYL07, YDHA07, PKK$^+$08, SP08]. The goal of large margin training is to assign significantly higher scores to correct transcriptions than competing ones; in particular, the margin between these scores is required to grow in proportion to the total number of recognition errors [SS07b, SS07a, PKK$^+$08, SP08, TGK04]. Empirically, large margin training has improved the performance of many systems beyond other leading discriminative approaches. We propose online updates that incrementally adapt the model parameters after a margin-based decoding of each training utterance. Comparing online versus batch implementations of large margin training, we find that the online methods converge more quickly. We also find that they yield acoustic models with better performance on phone recognition than other approaches—both online and batch—that do not attempt to enforce a large margin [CSS09b].

The third main contribution of this thesis is to study online updates that simultaneously transform the acoustic feature space used for ASR. Specifically, we show how to adapt the acoustic features computed by typical front ends in order to increase the margin of correct recognition. We adapt the feature space by learning highly dis-

criminative linear projections of acoustic features concatenated from multiple adjacent analysis windows. Optimizing the acoustic features in the front end presents new challenges for online learning. First, the optimization landscape becomes considerably more complex. Second, the projection matrix appears to be especially sensitive to the choice of learning rates. To deal with these difficulties, we explore many different schemes for initialization and parameter-tying [You92, DM94]. Interestingly, our best results are obtained by training several recognizers in parallel while tying the projection matrix used to compute acoustic features in their front ends. In our experiments, this form of parameter-tying *across different recognizers* yields consistent improvement beyond the already significant gains of large margin training.

We have published our preliminary explorations of these ideas in three previous studies [CSS09c, CSS09b, CSS09a]. In this thesis, we provide a unified presentation of these ideas and also include additional experiments on larger model sizes and different training paradigms.

1.3 Organization

The organization of this thesis is as follows. Part I of the thesis consists of Chapter 1 to 5, which provide background necessary to understand our main contributions. Chapter 2 reviews large margin methods for binary, multiway, and sequential classification, highlighting the different optimizations required for each of these settings. Chapter 3 reviews previous work on online learning algorithms, focusing especially on how these algorithms have been applied to large margin classification. Chapter 4 describes the use of hidden Markov models for ASR; we review the currently most popular approaches to discriminative training, as well as the major challenges posed by increasingly large training corpora. Chapter 5 provides more background on acoustic modeling in ASR, as well as benchmarks for current leadning approaches.

Part II of the thesis consists of Chapter 6 to 8, in which we explore the use of online algorithms for large margin training of HMMs. Chapter 6 introduces the basic form of our online updates for discriminative training of HMMs. Chapters 7 and 8 extend these updates to incorporate large margin constraints and to adapt the acoustic

feature space. Each of these chapters also presents experimental results on the TIMIT speech corpus; by interleaving results in this way, we hope to convey the evolution of ideas and practices that guided our own investigations.

Finally, in Chapter 9, we summarize our most important findings and discuss future directions for research.

Chapter 2

Large Margin Methods

In this chapter we review large margin methods on different types of classification problems. Large margin methods have been discussed in a number of literatures since the 1960s [DH73, Cov65], and among them, support vector machines (SVMs) are probably the most successful and widely used in pattern recognition. The initial form of SVMs was proposed in [BGV92] by using kernel functions to nonlinearly map input vectors to a high dimensional feature space and by constructing a maximal margin classifier in this space. Following the first paper, soft margin machines were introduced in [CV95]. Both of these methods can be carried out by efficient numerical algorithms. However, for various reasons, it is more challenging to apply SVMs to multiclass classification problems [CS01, SS06] and especially to sequence labeling problems [JH98, ATH03, SS07b]. We give an overview of SVMs in Section 2.1 and review other large margin methods and their extensions to multiway classification and sequence labeling problems in Sections 2.2 and 2.3.

2.1 Support vector machines

The generalization of a learning algorithm is controlled by two factors: the empirical risk, and the confidence interval which is determined by the model complexity [Vap95]:

$$R(f) \leq R_{emp}(f) + \Omega(f), \tag{2.1}$$

where R and R_{emp} are the actual risk and the empirical risk of a decision function f, and Ω is the confidence interval of the function. We can always find a classifier in a very high dimensional space (Ω increases) which separates the training data with zero error ($R_{emp} = 0$), but the generalization of the classifier may be poor. On the other hand, classifiers working with few free parameters or in a low dimensional space might not even obtain satisfactory error rates on the training data. There seems to be an inherent tradeoff between these factors. However, SVMs showed an elegant way to finesse this tradeoff: they keep the first term equal to zero by mapping input vectors to a very high dimensional space and minimize the second term in that space. The generalization error of SVMs is shown to be bounded by the number of support vectors, regardless of the dimensionality of the feature space.

SVMs provides state-of-the-art performance in many applications of pattern recognition. The key ingredients of SVMs are the maximal margin hyperplane and the kernel method: SVMs nonlinerly map the input vectors into a very high dimensional space, then find a maximal margin hyperplane in that space. A tight error bound is derived, and a convex optimization problem is formulated. All of these contribute to the success of SVMs.

The simplest setting of SVMs is a binary classification. Given a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \Re^d$ and $y_i \in \{-1, 1\}$. In binary classification, we seek a hyperplane to separate the positive and negative labeled data (Fig. 2.1). Let $f : \Re^d \rightarrow \Re$ be a linear discriminant function:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \tag{2.2}$$

where the label of an unknown example $\mathbf{x}$ is inferred from

$$g(\mathbf{x}) = \text{sgn}\left(f(\mathbf{x})\right). \tag{2.3}$$

For a hyperplane with a discriminant function $f(\mathbf{x})$, the *margin* is defined as the minimum distance from a correctly classified training point to the hyperplane; that is,

$$\rho = \min_i y_i f(\mathbf{x}_i) \geq 0 \quad \forall i \tag{2.4}$$

assuming that the training data are linearly separable. The margin ρ plays an important role in the generalization error bound in the SVM theory because it is related to the

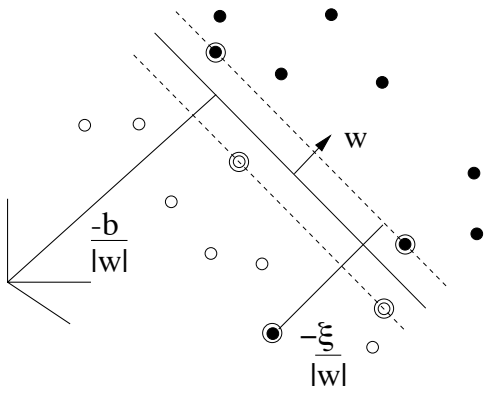

Figure 2.1: An illustration of the hyperplane constructed by SVM [Bur98].

VC-dimension [Vap82]. Thus, it is desirable to find a classifier f^* which maximizes the margin

$$f^* = \text{argmax}_f \rho_f = \text{argmax}_f \min_i y_i f(\mathbf{x}_i). \tag{2.5}$$

f^* is known as the *optimal hyperplane* [Vap82]. Without any constraint on the parameter $\mathbf{w}$, the maximum does not exist since one can always choose $\mathbf{w}$ to make the function value $f(\mathbf{w})$ arbitrarily large. However, by imposing a lower bound on $\mathbf{w}$ such that $\rho = 1$, the max-min problem (eq. (2.5)) can be transformed into an equivalent constrained optimization

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i, \end{aligned} \tag{2.6}$$

which represents the well-known quadratic programming problem of SVMs.

Another key element of SVMs is the kernel method. The basic idea of kernel methods is to imagine a nonlinear feature mapping $\phi : \mathcal{R}^d \rightarrow \mathcal{F}$ that maps the examples from the input space into a feature space $\mathcal{F}$. The dimension of the feature space can be very high, or even infinite, such that in this space we are much more likely to find a hyperplane that separates the training data. The direct computation for kernel methods is very expensive, since the dimension of the feature space is potentially infinite. Therefore, the so-called *kernel trick* is usually implemented, in which an implicit mapping of ϕ is induced by a kernel function. The kernel function defines the inner product of ex-

amples in the feature space, without an explicit computation of the nonlinear mapping, and the computation is typically implemented through a kernel matrix to facilitate the caching of the values. In order to incorporate the kernel trick, the objective functions have to be written in terms of inner products of examples. The constrained optimization task in eq. (2.6) can be formulated with respect to its dual variables by applying a Lagrangian. The optimization in the dual form remains a constrained quadratic problem in terms of dot products between input vectors, and thus makes it easy to apply kernel tricks in the computation.

SVMs are obtained by solving a problem in convex optimization - specially, a problem in quadratic programming (QP). The QP for SVM has many special properties, which can be exploited to develop more efficient solvers. In the early stage, SVMs were solved by standard techniques, such as interior point (IP) methods [BV04]. IP methods for SVMs are very accurate; however, their running time is cubic in the size of the training data N, and their memory requirement is $O(N^2)$. This complexity makes it difficult to use standard QP solvers for medium to large scale applications of SVMs.

The quadratic memory requirement comes from the assumption that the full kernel matrix is available. Computing a full kernel matrix is consuming in both time and space. To overcome this issue, many researchers have proposed decomposition methods [Pla99, Joa99]. Working on the dual representation of eq. (2.6), these methods break the full-scale optimization problem into a sequence of smaller, more manageable QP problems. Specifically instead of solving for all Lagrange coefficients at the same time, they select a *working set* of coefficients at each iteration and solve the corresponding subproblem for the working set. An extreme case of this approach is sequential minimal optimization (SMO) [Pla99], which selects two examples at a time to form a minimum subproblem. While working set methods are simple to implement, they still have a time complexity that is super linear in the training set size N [SSSS07].

In effect, all batch methods have a bottleneck in the complexity with respect to the training set size N. As the training set grows larger, the scalability of the SVM algorithm becomes an issue. Recently the demand for large scale applications has revived the interest in online learning and gradient based methods for SVMs. Many researchers have published works along this line [SSSS07, KSW01, Zha04]. It has even been shown

that a simple stochastic gradient descent (SGD) method can yield comparable results as a state-of-the-art solver [Bot07], but 10+ times as fast. These results have inspired our own exploration of online learning algorithms, especially for large margin methods, which we discuss in the next chapter.

So far, we have assumed that the training set is linearly separable. In practice, a separating hyperplane might not exist. Or, imagine if there are many noisy examples in the training data which cause a large overlap between different classes of data. In this case, the maximal margin hyperplane defined in eq. (2.6) may be entirely determined by the noisy examples. To overcome this sensitivity to the noise, we can relax the margin constraints and introduce a slack variable to measure how much they are violated [CV95]:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\textstyle\sum_i \xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i. \\ & \xi_i \geq 0 \end{aligned} \tag{2.7}$$

The loss function for so-called *soft margin* hyperplanes consists of two terms - one that regularizes the model complexity, and one that penalizes the margin violations. The slack variable ξ_i represents the distance from each example $\mathbf{x}_i$ to the margin, see Fig. 2.1. The problem in eq. (2.7) can also be written as the unconstrained optimization

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i [1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)]^+, \tag{2.8}$$

where $[z]^+ = \max(0, z)$ is the so-called *hinge function*. Unconstrained optimizations generally converge more quickly than constrained optimizations, making them more suitable for gradient based methods.

2.2 From binary to multiclass

So far we have discussed the use of SVM for problems in binary classifications. However, many real world problems have more than two classes, and there have been many attempts to generalize SVMs to this setting [PCSt99, WW99, CS00, CS01]. In

this section we briefly review how large margin classifiers have been applied in this setting.

The goal of multiway classification is to map an unknown input $\mathbf{x} \in \Re^d$ to a set of predefined classes $c \in \{1, 2, \ldots, C\}$, where d is the input dimensionality and C is the number of classes. Multiclass learning algorithms with SVMs have been discussed by many authors. The classical approach in [SBV95] considers the multiclass problem as a collection of binary classification problems. In the one-versus-rest approach, C binary classifiers are constructed, where each classifier uses a hyperplane to separate one class from the other $C - 1$ classes. An input is assigned to the class with the maximum positive distance from the input to the separating hyperplane for that class. An alternative approach is the one-versus-one method, which constructs $C(C + 1)/2$ classifiers for separating each pair of competing classes. The overall classification is then made by a voting method that combines these pairwise decisions to choose the most likely label. Despite the simplicity and effectiveness of this framework, it cannot capture the correlations between classes since each binary classifier is trained independently of all the others.

The previously mentioned approaches decompose the multiway classification problem into multiple independent binary classification problems. However, it is also possible to train multiclass SVMs with a single optimization [WW99, CS01, ASS00]. To do so, a piecewise linear classifier is constructed for each of the C classes, and the overall decision rule is given by:

$$g(\mathbf{x}) = \text{argmax}_c \left[(\mathbf{w}_c \cdot \mathbf{x}) + b_c) \right], \quad c = 1, \ldots, C. \tag{2.9}$$

There are two important formulations for these types of multiclass SVMs [WW99, CS01]. Weston et al. [WW99] proposed to find an optimal hyperplane which yields the margin separating the target class from all other classes. In this approach, the binary classification is extended by generalizing the constraints in eq. (2.6) to guarantee a margin from each example to the yperplane for all competing classes. Therefore, the number of constraints becomes $N \times C$, which grows proportional to the number of classes in the classification problem. The required optimization is a single QP, but it is correspondingly harder to solve than the original one for binary SVMs.

Crammer et al. [CS01] proposed a slightly different formulation. These authors

defined the loss of a classifier as the difference between the discriminant score of the target class and that of the inferred class. The difference between these two works is that Crammer et al. only penalized the most incorrect (or most nearly incorrect) decision boundary , while Weston et al. penalized all competing class boundaries which grow proportionally to the number of classes. In the work of Crammer et al., the constrained optimization is given by

$$\begin{aligned} \min \quad & \tfrac{1}{2}\textstyle\sum_c \|\mathbf{w}_c\|^2 + \gamma \sum_i \xi_i^2 \\ \text{s.t.} \quad & \max_c\{\mathbf{w}_c \cdot \mathbf{x}_i + 1 - \delta_{y_i,c}\} - \mathbf{w}_{y_i} \cdot \mathbf{x}_i \leq \xi_i, \quad \forall i \end{aligned} \tag{2.10}$$

where c denotes the index of classes. Here, $\delta_{y_i,c}$ denotes the indicator function which returns 1 if $y_i = c$ and zero otherwise. In this equation, for simplicity we have dropped the bias b_c for each hyperplane; biases can be incorporated in the optimization with a little extra care [SSSS07]. The optimization is most easily solved by transforming it into its dual form with N unknowns (one for each training example). The authors develop a specialized solver by decomposing this dual QP into N smaller QPs (with C constraints in each) and applying iterative methods to solve the small QPs. A number of tricks are used to accelerate the computation, including greedy example selection and kernel values caching. Multiclass SVMs based on eq. (2.10) have certain advantages over collections of independently trained binary SVMs; however, the computation of independently trained binary SVMs might be more efficient [RK04].

Notwithstanding the above successes, it remains challenging to apply traditional SVMs to multiclass problems, especially large-scale applications. There are two main issues. First, in order to obtain a nonlinear decision boundary in the input space, a kernel matrix must be constructed with as many rows and columns as the training examples. This generally necessitates the use of subset methods, as well as the caching of kernel values. Second, the complexity grows with the number of classes - either linearly or quadratically - depending on how the binary classification is generalized to multiway classification.

Another line of research in parallel to SVMs has suggested an alternative way to generalize large margin classifiers to multiway classification. In [SS06], a large margin classifier based on Gaussian mixture models (GMMs) is introduced. The class boundaries defined by GMMs are innately nonlinear in the input space; therefore, in these large

margin classifiers, no kernel method is needed to obtain a nonlinear decision boundary.

We begin by reviewing these GMMs with only single Gaussian component per class. The inference in large margin GMM is done by computing the minimum Mahalanobis distance among those from the example $\mathbf{x}$ to class centroids $c = \{1, \ldots, C\}$:

$$g(\mathbf{x}) = \text{argmin}_c\{(\mathbf{x} - \mu_c)^T \Sigma_c^{-1} (\mathbf{x} - \mu_c) + \theta_c\}. \tag{2.11}$$

In eq. (2.15), the parameters μ_c and Σ_c are the mean and covariance matrix of the Gaussian model for class c, and θ_c is a scalar offset for calibrating the scores of different classes.

In order to simplify the representation of the decision boundary (analogous to linear SVMs), [SS06] reparameterized the Gaussian density function by an enlarged parameter matrix Φ:

$$\Phi = \begin{bmatrix} \Sigma^{-1} & -\Sigma^{-1}\mu \\ -\mu^\top \Sigma^{-1} & \mu^\top \Sigma^{-1} \mu + \theta \end{bmatrix}. \tag{2.12}$$

The decision function in eq. (2.15) can be rewritten in terms of this new parameterization as:

$$g(\mathbf{x}) = \text{argmin}_c\{\mathbf{z}^T \Phi_c^{-1} \mathbf{z}\}, \text{ where } z = \begin{bmatrix} x \\ 1 \end{bmatrix}. \tag{2.13}$$

As in eq. (2.9) for multiclass SVMs, the discriminant function has a piecewise linear dependence on the classifier parameters. Note how the Mahalanobis distances are defined in terms of the positive semidefinite matrices Φ_c.

Inspired by SVMs, large margin GMMs also attempt to separate the data from the decision boundaries of competing classes by a large margin. Fig. 2.2 shows the decision boundary defined by large margin GMMs (for two classes), which is an ellipsoid rather than a half-plane (as in binary SVMs). The constraints in large margin GMMs are given by:

$$\mathbf{z}_i^T \Phi_c \mathbf{z}_i \geq 1 + \mathbf{z}_i^T \Phi_{y_i} \mathbf{z}_i, \quad \forall c \neq y_i. \tag{2.14}$$

where $\mathbf{z}$ is defined as in eq. (2.13). Large margin GMMs are trained by relaxing these margin constraints via slack variables ξ_i and solving the constrained optimization:

$$\begin{aligned} \min \quad & \textstyle\sum_c \text{trace}(\Phi_c) + C \sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{z}_i^T \Phi_c \mathbf{z}_i - \mathbf{z}_i^T \Phi_{y_n} \mathbf{z}_i \geq 1 - \xi_i, \quad \forall i, c. \end{aligned} \tag{2.15}$$

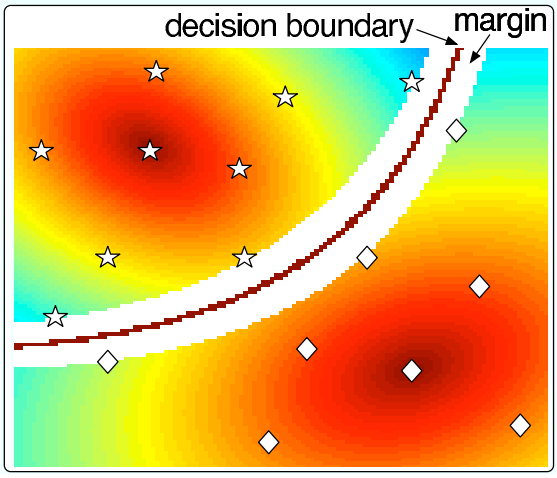

Figure 2.2: An example of the nonlinear hyperplane constructed by large margin GMM [SS07b].

This formulation can also be extended by replacing the single Gaussian log-likelihood for each class c by the log-likelihood of a Gaussian mixture model for each class.

Eq. (2.15) is a convex optimization which can be solved by general interior point methods. However, interior point methods for this problem do not scale well to large training sets. In practice, the optimization is performed by a specialized gradient-based method. This is done by rewriting the optimization and incorporating the constrains via a hinge function. Specifically, we consider the loss function:

$$\mathcal{L} = \sum_c \text{trace}(\Phi_c) + \gamma \sum_i [\mathbf{z}_n^T \Phi_{y_n} \mathbf{z}_n - \mathbf{z}_n^T \Phi_c \mathbf{z}_n + 1]^+. \tag{2.16}$$

The loss function in eq. (2.16) can be easily minimized by gradient-based methods for *unconstrained* optimization.

2.3 Sequential classification

So far we have discussed classification for i.i.d. data. Data correlated in time or in space give rise to problems in sequential classification. Applications of sequential classification include natural language processing, speech recognition, computational

biology, and internet data analysis. The most popular models for sequential classification are hidden Markov models (HMMs).

The goal of sequential classification is to map a sequence of unknown inputs $\{\mathbf{x}_t\}_{t=1}^T \in \Re^d$ into a sequence of labels $\mathbf{y}$ with C^T possible outputs, where T is the length of the input sequence and C is the number of possible classes at each time t. This goal is not easily accommodated by large margin classifiers such as SVMs. Recall that the training complexity of SVMs increases at least linearly with the number of classes. In the sequential classification, the number of classes grows exponentially with the sequence length. This exponential growth makes it difficult to naively apply multiclass SVM algorithms to these problems. It remains an active research area to construct large margin classifiers that incorporate the temporal correlations of sequential examples [ATH03, JLL06a, SS07b, JFY09].

Following the work by Crammer et al. [CS01], Altun et al. [ATH03] proposed the first formulation for combining a large margin classifier and an HMM. The hidden Markov support vector machine (HM-SVM) first defines the feature mapping $\phi(\mathbf{x}, \mathbf{y})$ to map an input sequence $\mathbf{x}$ and its label sequence $\mathbf{y}$ to an extended feature space. Inspired by HMM, HM-SVMs include two types of features. The first type describes attributes of the given label; the second type describes the interaction between neighbouring labels. The inference is done by

$$\mathbf{f}(\mathbf{x}) = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}), \tag{2.17}$$

where

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w} \cdot \phi(\mathbf{x}, \mathbf{y}) = \sum_t F(\mathbf{x}, \mathbf{y}; t) \tag{2.18}$$

can be viewed as the summation of individual likelihoods and state transition likelihoods given a sequence $\mathbf{x}$. The feature mapping is designed to ensure that the inference can be performed by Viterbi-decoding like algorithms. The feature mapping can also be implicitly defined by specifying a kernel K on the input space

$$K((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \phi(\mathbf{x}, \mathbf{y}) \cdot \phi(\mathbf{x}', \mathbf{y}'). \tag{2.19}$$

The constrained optimization in HM-SVMs can be written as

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & F(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y}\neq\mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y}) \geq 1 \quad \forall i. \end{aligned} \tag{2.20}$$

[ATH03] performed this optimization with a Perceptron-like algorithm. There are two important contributions of this work: 1) although the algorithm works on sequential data, the number of constraints remains linear in the sequence length; 2) the algorithm imposes few limitations on the data and works well for large feature sets and long sequences. Note, however, that the model used in the paper is essentially a discrete hidden Markov model; it is unclear how to apply the algorithm to classify sequences of continuous feature vectors such as those that occur in automatic speech recognition.

However, if a kernel learning setting is desired in order to benefit from the sparseness in the feature space, the dual representation of eq. (2.20) has to be investigated, and the constraints have to be expanded to represent all possible hypotheses. In this case, the number of constraints grows exponentially as the length of the data.

Taskar et al. [TGK04] proposed a similar framework for structure learning based on large margin methods. The max-margin Markov networks (M^3 nets) differs from HM-SVM by two distinctions. The first one is its incorporation of a variable-sized margin, instead of the $0-1$ loss as in eq. (2.20). In particular, the optimization problem is formulated as

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & F(\mathbf{x}_i, \mathbf{y}_i) - F(\mathbf{x}_i, \mathbf{y}) \geq \Delta(\mathbf{y}_i, \mathbf{y}) \quad \forall i, \quad \forall \mathbf{y}, \end{aligned} \tag{2.21}$$

where $\Delta(\mathbf{y}_i, \mathbf{y})$ denotes the *per-label* loss between the ground-truth sequence $\mathbf{y}_i$ and the hypothesis $\mathbf{y}$. The variable-sized margin makes the optimization more dedicated to the sequential classification applications, and can be generalized to minimize other loss functions used in the evaluation.

The second distinction is their simplification of the constraints in eq. (2.21) based on the assumption of a chain Markov network. They replace the exponential constraints by constraints over cliques and each node, which results in a polynomial-sized formulation. Eq. (2.21) is then optimized by the SMO algorithm in dual representations. Note, however, that both HM-SVM and M^3 nets are developed for discrete sequences, such as handwriting recognition and part-of-speech tagging.

Unlike the applications with discrete sequences, hidden Markov models and Gaussian mixture models are the most widely used for modeling continuous data, especially for applications of ASR. While generative models such as HMMs have achieved

a good performance in modeling speech data, discriminative models have yielded state-of-the-art performances in the recent developments. The emergence of large margin methods in machine learning, especially on the nonlinear models, have also drawn a great deal of interest in the speech community [LYL07, SS07b, JLL06b, KSSSC07].

Motivated by applications in ASR, Sha et al. [SS07b] proposed an extension of large margin GMMs for problems in sequential classification. For these models, they first define the discriminant function as the logarithm of the continuous-density HMM

$$\mathcal{D}(\mathbf{x}, \mathbf{s}) = \sum_t \log a(s_{t-1}, s_t) - \sum_t z_t^t \Phi_{s_t} z_t, \tag{2.22}$$

where Φ_{s_t} is defined as in large margin GMMs. The output space can be viewed as the space containing all combinations of label sequences $\mathbf{y}$, which is combinatorial in the sequence length T. Inference is performed by Viterbi decoding in this space:

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} \mathcal{D}(\mathbf{x}, \mathbf{y}). \tag{2.23}$$

As in large margin GMMs, large margin HMMs seek to separate the data $(\mathbf{x}_i, \mathbf{y}_i)$ from all competing classes/sequences $\mathbf{y}$ by a large margin:

$$\mathcal{D}(\mathbf{x}_i, \mathbf{y}_i) - \mathcal{D}(\mathbf{x}_i, \mathbf{y}) \geq \mathcal{H}(\mathbf{y}_i, \mathbf{y}), \quad \forall i, \mathbf{y} \in \mathcal{Y}, \tag{2.24}$$

where $\mathcal{H}$ denotes the *Hamming distance* between the two sequences, and $\mathcal{Y}$ is the combinatorial output space of the label sequences. To handle the exponentially many constraints in eq. (2.24), the algorithm collapses them into a single convex constraint using a *softmax* inequality, while keeps the problem convex; see [SS07b] for details.

The resulting optimization is given by:

$$\begin{aligned} \min \quad & \textstyle\sum_c \text{trace}(\Phi_c) + C \sum_i \xi_i \\ \text{s.t.} \quad & \log \textstyle\sum_{\mathbf{s} \neq \mathbf{y}_i} e^{\mathcal{H}(\mathbf{y}_i, \mathbf{y}) + \mathcal{D}(\mathbf{x}_i, \mathbf{y})} - \mathcal{D}(\mathbf{x}_i, \mathbf{y}_i) \leq \xi_i, \quad \forall i, c \\ & \Phi_c \geq 0 \quad \forall c. \end{aligned} \tag{2.25}$$

To perform this optimization, [SS07b] developed a special-purpose solver that first computes a good initialization using conjugate gradient methods, then refines the results by a projected subgradient method.

A potential shortcoming of all the methods in this section is that they are *batch* methods for sequential classification. For very large data sets, as occur in ASR, it may not be possible to store all the training sequence in memory. This motivates our discussion of online methods in the next chapter of the thesis.

Chapter 3

Online Learning

The history of online learning can be traced back to early learning systems, which required simple algorithms due to the limitations of the hardware. However, today's large datasets have led to a resurgence of interest in online algorithms, especially for large-scale applications such as automatic speech recognition (ASR). While the first online learning algorithms emerged from applications focused on simple settings - such as binary classification with linear models - these algorithms have now been extended to a wide range of applications, including multiway classification [SS99, CS03, KSST08] and sequence labeling problems [Col02, MCP05, MP06]. In this chapter we review previous work in online learning, focusing especially on work related to large margin methods and large-scale applications.

Online algorithms usually work in rounds. In each round, the algorithm sees an example and updates the parameters to improve its model. Online algorithms differ from batch algorithms which accumulate statistics over an entire training set and update model parameters after each such pass. Online algorithms emerged to compensate for the main disadvantages of batch algorithms (e.g., the need to store large amounts of training data in memory, the massive computation required per model update), which can hinder large-scale applications [BL04].

3.1 Perceptron

The simplest and the most fundamental online algorithm is perhaps Rosenblatt's Perceptron [Ros58]. The perceptron is an algorithm for binary classification. Given a sequence of examples $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, the goal of binary classification is to map an unknown observation $\mathbf{x}$ to a class label $c = \{+1, -1\}$. Here we assume that the training examples are received in a sequential manner; namely, at time *t*, the incoming example is $(\mathbf{x}_t, y_t)$.

The perceptron attempts to learn a linear discriminant function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The decision rule $g(\mathbf{x})$ for the perceptron is given by

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } f(\mathbf{x}) > 0, \\ -1 & \text{if } f(\mathbf{x}) < 0. \end{cases}, \tag{3.1}$$

Whenever a misclassification occurs, the parameter $\mathbf{w}$ is updated by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t\mathbf{x}_t. \tag{3.2}$$

The algorithm iterates through the whole training set until there are no more misclassifications (if the examples are linearly separable) or until the error rate is as small as possible (if they are not).

In addition to its extreme simplicity, the perceptron provably converges to an error-free hyperplane in a finite number of iterations when the data is linearly separable [FS99]. However, convergence does not occur if the examples are not linearly separable. In this case, because the learning rate of the basic perceptron is fixed, the algorithm may exhibit severe fluctuations.

Although the basic perceptron cannot handle non-linearly separable data [Bis96], extensions of the algorithm have been proposed to cover this case (e.g., choosing a non-linear feature mapping [CV95, SB00], or replacing the linear model by a deep architecture [Bis96]).

When the data is linearly separable, the perceptron algorithm will eventually converge to the optimal separating hyperplane. However, the data might be non-linearly separable or we do not want to wait until the convergence. When this happens, the question is how to force the perceptron algorithm to converge to a single "nearly optimal"

solution. One solution could be using a decreasing learning rate η_t, which is originally suggested in the context of stochastic gradient descent (SGD) in the backpropagation for neural network training [LBOM98]. Another solution, provided by [FS99], is to use the classifier which has survived for the longest time in the online updates. The more training examples the perceptron correctly classifies, the more likely it is the perceptron may also correctly classify an unknown data.

The alternative version of the perceptron algorithm seeks an optimal classifier from the sequence of classifiers generated in the online learning without the assumption of linear separability. The algorithm in [FS99] works by counting the number of survivals of every perceptron generated in the online learning, and the final prediction is *voted* by all these perceptrons, which they call a "*voted perceptron*". Given a list of weighted perceptrons $\{(f_1, c_1), (f_2, c_2), \ldots, (f_k, c_k)\}$, in which c_i is the weight of the ith perceptron f_i, the voted perceptron computes the prediction for an unknown input $\mathbf{x}$ by

$$\hat{y} = \text{sign}(\sum_{i=1}^{k} c_i \text{sign}(f_i(\mathbf{x}))). \tag{3.3}$$

The weights c_i are determined by how long one hypothesis f_i survives in the training examples. The voted perceptron attempts to model the data that are not linearly separable or to accelerate convergence to an optimal hyperplane (which in general takes a long time) even when the data are linearly separable. Intuitively, the voted perceptron looks for good hypotheses by examining which simple perceptrons survive the longest time. Then, in the final voting, these good hypotheses are assigned heavier weights. Geometrically, the optimal hypothesis is estimated by constructing a weighted average of all the suboptimal hypotheses based on the survivals during training. [FS99] derived a generalization bound for the voted perceptron under the condition that the data are almost linearly separable.

The experiments in [FS99] show that this alternative algorithm for training large margin classifiers converged faster but generalized poorer than traditional SVMs on MNIST dataset. Nevertheless, the voted perceptron helped the convergence by a significant amount than the standard perceptron, especially in the non-linearly separable condition (i.e., in the low dimensional feature space).

In the implementation, the voted perceptron has to store every single updated

perceptron and its count of survivals. This prevents the voted perceptron being scaled to a larger training set or unlimited data. An equivalent method, *average perceptron*, is more memory efficient and exhibits the same theoretical and experimental properties as the voted perceptron [FS99]:

$$\hat{y} = \text{sign}(\sum_{i=1}^{k} c_i f_i(\mathbf{x})). \tag{3.4}$$

The average perceptron has been adopted in many perceptron-like algorithms to stabilize the fluctuations in the online updates [Col02, Zha04].

3.2 Multiclass Perceptron

We have discussed the extension from binary classification to multiway classification for large margin methods in Section 2.2. In a similar way, we can generalize the original perceptron to handle multiclass problems [DH73, CS03, FSSSU06].

The goal of multiway classification is to infer the class label $c \in \{1, 2, ..., \mathcal{C}\}$ of an unknown observation $\mathbf{x}$ from the training set $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ with N observations. Let $\mathbf{w}_j$ indicate the weight vector associated with the jth class. The goal of the multiclass perceptron is to learn weight vectors such that

$$\mathbf{w}_{y_i}^T \mathbf{x}_i \geq \max_j \mathbf{w}_j^T \mathbf{x}_i, \quad \forall(\mathbf{x}_i, y_i). \tag{3.5}$$

In other words, it is desired that the *score* of the correct class is greater than those of all competing classes.

The optimization problem is to ensure that eq. (3.5) is satisfied for as many training examples as possible. A mistake-driven update can be developed for multiclass perceptrons analogous to the one in eq. (2.9). Specifically, let $\hat{y}_i$ denote the predicted label of the training example $\mathbf{x}_i$, given by:

$$\hat{y}_i = \text{argmax}_j(\mathbf{w}_j^T \mathbf{x}_i). \tag{3.6}$$

Then, if a mistake is made ($\hat{y}_i \neq y_i$) on this example, the parameters are updated by

$$\begin{aligned} \mathbf{w}_{y_i} &\leftarrow \mathbf{w}_{y_i} + \eta \mathbf{x}_i \\ \mathbf{w}_{\hat{y}_i} &\leftarrow \mathbf{w}_{\hat{y}_i} - \eta \mathbf{x}_i, \end{aligned} \tag{3.7}$$

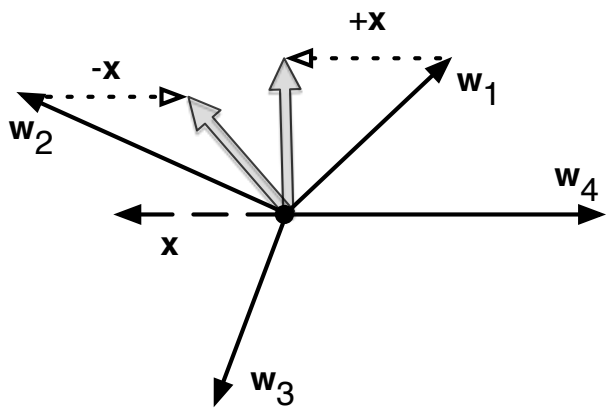

Figure 3.1: An illustration of multiclass perceptrons for eq. (3.7). Suppose the training example $\mathbf{x}$ of class 1 is misclassified by class 2, both weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are adjusted by an amount proportional to $\mathbf{x}$.

where η is a user-defined learning rate. The update rule is geometrically illustrated in Fig. 3.1.

Multiclass perceptrons share the same basic features as binary perceptrons. They are simple to implement and provably converge when the data is linearly separable. When the data is not separable, convergence to a "nearly optimal" solution can be obtained by decaying the learning rate and/or averaging the results over time, as discussed earlier for binary perceptrons.

The multiclass perceptron algorithm was derived as a special case of so-called *ultraconservative* learning algorithms in [CS03]. At each iteration, these algorithms update the weight vectors for all classes with higher overlaps than the target class. In particular, let $E = \{r \neq y_i | \mathbf{w}_r \cdot \mathbf{x}_i \geq \mathbf{w}_{y_i} \cdot \mathbf{x}_i\}$ denote the error set for example $(\mathbf{x}_i, y_i)$. When $E \neq \phi$, the update rules are given by:

$$\begin{aligned} \mathbf{w}_{y_i}^{t+1} &\leftarrow \mathbf{w}_{y_i}^{t} + \mathbf{x}_i, \\ \mathbf{w}_{r}^{t+1} &\leftarrow \mathbf{w}_{r}^{t} - \frac{1}{|E|}\mathbf{x}_t, \quad \forall r \in E. \end{aligned} \tag{3.8}$$

The number of errors made by the classifier is invariant to scale transformations $\mathbf{w}_i \rightarrow \lambda \mathbf{w}_i$ of all the weight vectors. However, by imposing a minimum norm constraint on all the weight vectors, a unique solution is obtained. In fact, by incorporating a large

margin constraint, eqs. (3.7-3.8) can be viewed as a form of stochastic gradient descent on the loss function for multiclass SVMs.

3.3 Structured Perceptron

The perceptron algorithm has also been generalized to sequential classifications [Col02]. It was first applied in this way to improve part-of-speech tagging by hidden Markov models (HMMs). In this thesis we review the structured perceptron algorithm for sequential classification in more general terms.

Given a set of training data $\{(\mathbf{x}^i_{[1:n_i]}, \mathbf{y}^i_{[1:n_i]})\}_{i=1}^N$, where $\mathbf{x}^i_{[1:n_i]}$ is the ith observation sequence of length n_i and $\mathbf{y}^i_{[1:n_i]}$ is its true label sequence, our goal is to learn a weight vector $\mathbf{w}$ such that we can accurately predict the label $\hat{\mathbf{y}}$ of any observation $\mathbf{x}$ as

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}). \tag{3.9}$$

Here $\phi(\mathbf{x}, \mathbf{y})$ maps the observation/label sequence to a set of features which encode important correlations. For example, in part-of-speech tagging, the feature may count the co-occurrences of certain words, tags, and histories. More generally, the feature vector $\phi(\mathbf{x}, \mathbf{y})$ counts the number of emissions and state transitions over the whole sequence. The weight vector in eq. (3.9) plays a role analogous to the (log) emission and state transition probabilities in discrete HMMs. The optimization in eq. (3.9) can be performed by dynamic programming algorithms such as Viterbi decoding.

Assuming the data are linearly separable, we can look for a weight vector $\mathbf{w}$ such that

$$\mathbf{w}^T \phi(\mathbf{x}^i_{[1:n_i]}, \mathbf{y}^i_{[1:n_i]}) \geq \max_{\mathbf{y}} \mathbf{w}^T \phi(\mathbf{x}^i_{[1:n_i]}, \mathbf{y}_{[1:n_i]}), \quad \forall i. \tag{3.10}$$

More generally, we can attempt to minimize the difference between the left and right terms in the above equation whenever an error does occur. As in the original perceptron algorithm, this can be done by updating the weight vector in an online fashion. For sequential classification, the mistake-driven update is given by:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \eta(\phi(\mathbf{x}^i_{[1:n_i]}, \mathbf{y}^i_{[1:n_i]}) - \phi(\mathbf{x}^i_{[1:n_i]}, \hat{\mathbf{y}}^i_{[1:n_i]})), \tag{3.11}$$

which is performed whenever an error occurs (i.e., $\mathbf{y}^i \neq \hat{\mathbf{y}}^i$).

Table 3.1: A summary of Perceptron-based classification for different applications.

Perceptron-based updates	
Binary	$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta y_t \mathbf{x}_t$
Multiclass	$\mathbf{w}_{y_i} \leftarrow \mathbf{w}_{y_i} + \eta \mathbf{x}_i$ $\mathbf{w}_{\hat{y}_i} \leftarrow \mathbf{w}_{\hat{y}_i} - \eta \mathbf{x}_i$
Structured	$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \eta(\phi(\mathbf{x}^i_{[1:n_i]}, \mathbf{y}^i_{[1:n_i]}) - \phi(\mathbf{x}^i_{[1:n_i]}, \hat{\mathbf{y}}^i_{[1:n_i]}))$

For part-of-speech tagging, this update rule has a simple intuition. Suppose that $\phi(\mathbf{x}^i_{1:n_i}, \mathbf{y}^i_{1:n_i})$ counts the number of co-occurrences of particular words and tags. If a unique word in the sequence $\mathbf{x}^i$ was incorrectly tagged in the decoded label sequence $\hat{\mathbf{y}}^i$, then the update rule would increase (by $+\eta$) the corresponding weight for the correct word-tag count and decrease (by $+\eta$) the corresponding weight for the incorrect word-tag count. A table of the perceptron algorithms discussed in this chapter is summarized in Table 3.1.

3.4 Online learning in large margin methods

The main difficulty in SVM training arises from the large quadratic programming problem (eq. (2.6)). With as many constraints as labeled examples, the primal problem becomes unmanageable for large training sets. For this reason, most state-of-the-art SVM solvers work in the dual space of the SVM optimization problem (see Chapter 2). The dual has two advantages: 1) it lends itself naturally to the kernel tricks, and 2) the solution is sparse, enabling active set methods that focus on a subset of training examples. These so-called "decomposition methods" or "subset methods" are based on incremental updates. Although decomposition methods have greatly reduced the training times for SVMs, the amount of computation still scales super linearly in the number of labeled examples; also, these algorithms do not have known asymptotic rates of convergence (see Section 2.1). Thus, it remains an ongoing research problem to find the most efficient way to solve the quadratic programming problem in SVMs.

The above issues have inspired a number of online learning algorithms for computing maximal margin hyperplanes, including perceptron-like approaches [SSSS07, CDSSS03, CKS03, LL02]. The first work in this area was done by Li et al [LL02],

whose Relaxed Online Maximum Margin Algorithm (ROMMA) converted the batch optimization in SVM to an online task. Recall that the constraints in SVM optimization eq. (2.6) form a convex polyhedron with one constraint per labeled example. For online learning, this suggests that at each iteration t, the hypothesis $\mathbf{w}_t$ must be chosen to satisfy the constraints generated by all previous examples. ROMMA relaxes this requirement by replacing the polyhedral constraint with two simpler linear inequalities. With this relaxation, the mistake-driven update rule reduces to solving a least square problem. On the MNIST data set, ROMMA yielded better results than voted perceptrons (but wirse results than SMO, an active set method for solving the batch optimization in SVMs).

[Bot07] has suggested that online learning algorithms may obtain comparable results as batch algorithms in less training time. The most straightforward online algorithms for SVMs applies stochastic gradient descent (SGD) to the unconstrained optimization problem in eq. (2.16). A number of researchers employ this idea. [Zha04, KSW01] adapted SGD to the SVM optimization (similar to eq. (2.16)) with a sophisticated tuning of the regularization term and learning rate. More recently, Shalev-Schwartz et al. [SSSS07] developed a more sophisticated solver, which attempts to combine the advantages of online and batch methods. They use a subgradient method on mini-batches of training examples, decomposing the training data into chunks and performing online updates on these small subsets of examples. All these algorithms can be combined with kernel methods. For linear SVMs, a simple SGD solver has been shown to obtain comparable results (on the MNIST dataset) as state-of-the-art batch implementations, such as SVM-Lite [Joa99].

As discussed earlier, an important key to the success of SVMs is the use of kernel methods. By mapping the input vectors into an arbitrary high dimensional feature space, these methods ensure that the data becomes linearly separable, or at least nearly so. The feature mapping is defined implicitly through the kernel function, which is then used to compute the kernel matrix of inner products between training examples. In the dual formulation of SVMs, the training examples appear only through these inner products [Bur98].

However, kernel-based online algorithms are in general working in the primal formulations (e.g. the algorithms in [LL02] and [FS99]). When an error occurs, the

weight vector is updated by adding a component in the direction of the mistaken training example:

$$\mathbf{w}_i = \sum_{j=1}^{i} y_{\varepsilon_j} \mathbf{x}_{\varepsilon_j}, \tag{3.12}$$

where ε_j is the index to the training data which makes an error in the jth update, also known as *support vectors*. According to eq. (3.2), the decision rule can be represented in terms of dot products between the support vectors and the unknown input, which is of the general form to apply kernel tricks.

The weight vector in eq. (3.12) is represented by a combination of the erroneous instances, which must be stored as a *support set*. It is clear that if the data is not linearly separable, then the updates will never stop, requiring unbounded memory. Several researchers have addressed this problem in the past. Crammer et al [CKS03] proposed the first online algorithm to overcome the unbounded accumulation of support vectors: they control the memory requirement by setting a "budget" on the support set and pruning redundant support vectors whenever new candidate support vectors are added to the set. There are many open issues in this line of research, such as how to identify redundancy and how to prune support vectors. These issues have also been studied by [DSsS08, OKC08, DCP08].

Chapter 4

Automatic Speech Recognition

Research in automatic speech recognition (ASR) began several decades ago and still remains active. The earliest techniques were based on our understanding of human speech production and perception, and attention was focused on small-vocabulary applications, such as isolated word recognition. In the 1970's, the field of ASR was revolutionized by statistical methods that modeled the density of the speech signal, as represented by a sequence of acoustic feature vectors. Over time, hidden Markov model (HMM) have emerged as the leading statistical framework for ASR [JR05].

Today we face difficult challenges in large-vocabulary speech recognition, multilingual transcription, and speech-to-speech translation [PP02, GY08]. Some of these tasks have already been commercialized, owing to the high accuracy of current ASR systems. This accuracy (though still not perfect) has been due to steady advances in acoustic modeling and statistical estimation. In this chapter we review the current foundation of ASR, with a special focus on recent advances in acoustic modeling.

4.1 Basic architecture of a speech recognizer

The input to a continuous speech recognizer is a speech waveform recorded from a microphone. Typically, the speech signal is analysed every 10ms with a sliding window of 25ms in length. Based on this analysis, the front end of the recognizer segments and converts the waveform into a sequence of acoustic feature vectors. Finally, the acoustic feature vectors are fed into a decoder, which is the heart of a recognizer. The

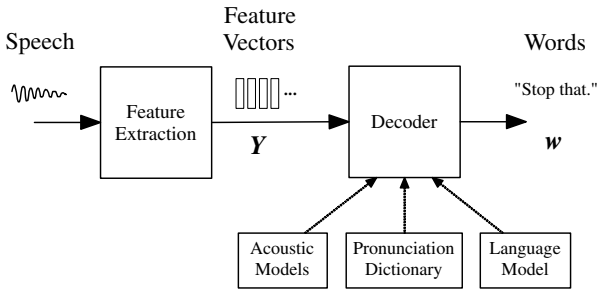

Figure 4.1: An overview of the architecture of a modern speech recognizer [GY08].

process is illustrated in Fig. 4.1.

Given a sequence of acoustic feature vectors $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, the goal of the decoder is to find the sequence of *words* $\mathbf{W}$ that generated the utterance. The inference is done by computing *maximum a posteriori* (MAP) word sequence:

$$\hat{\mathbf{W}} = \text{argmax}_{\mathbf{W}} p(\mathbf{W}|\mathbf{X}) = \text{argmax}_{\mathbf{W}} p(\mathbf{W}) p(\mathbf{X}|\mathbf{W}). \tag{4.1}$$

Note that the length of the word sequence can vary from hypothesis to hypothesis in a continuous speech recognizer; the best length is determined by the MAP inference of $\hat{\mathbf{W}}$ in eq. (4.1).

The first term $p(\mathbf{W})$ in eq. (4.1) is the recognizer's *language model*, while the second term $p(\mathbf{X}|\mathbf{W})$ is the recognizer's *acoustic model*. The language model estimates the probability of word transitions in a given language. It is usually represented by an N-gram model, where N is an integer. Namely, the probability of a word W_k appearing in a sentence depends on the preceding $N-1$ words:

$$P(W_k|W_1, W_2, ..., W_{k-1}) = P(W_k|W_{k-n+1}, ..., W_{k-1}). \tag{4.2}$$

The language model can be learned by counting the co-occurrence frequencies of words in text documents. The function of a language model is to prune the search space of hypotheses during decoding. If a sentence is unlikely to appear in one language, the decoder will eliminate this sentence based on its small word transition probability. To define the space of hypotheses, modern ASR systems usually assume a predefined vocabulary. A dictionary lists all the words in the vocabulary, as well as specifying their constituent phonemes (i.e., the smallest units of speech that are used in a particular language).

All of the blocks in Fig. 4.1 are customized and integrated to optimize the performance of the decoder. Next, we review the modules for acoustic feature extraction and acoustic modeling.

4.2 Acoustic feature extraction

The front end of the recognizer typically analyses the speech signal in sliding, overlapping windows of length ~ 25ms. A new frame of acoustic features is created by sliding these windows in intervals of ~ 10ms over the speech signal. The duration of each frame is so short that the spectral information within each frame is assumed stationary. The spectral information is computed by a Fourier transform or linear prediction analysis [GY08].

A number of different feature extraction techniques have been studied in ASR [Her90]. Here we review the computation of Mel-frequency cepstral coefficients (MFCCs). To compute MFCCs, the Fourier spectrum is first smoothed by a set of band-pass filters that are uniformly spaced on a *Mel-scale* of the frequency spectrum. The Mel-scale is inspired by human listeners, whose frequency selectivity is linear for frequencies under 1000Hz and logarithmic for higher frequencies. The filter outputs are passed through a log-compression in order to make them approximately Gaussian, then decorrelated by a discrete cosine transform (DCT). Alternatively, similar decorrelation effects can be obtained by Linear Prediction method or KL transform [Gal99]. The purpose of decorrelation is to enable the low-order coefficients of the DCT to be modeled as independent Gaussian variables. Usually the first 13 coefficients of the DCT are retained as the so-called cepstral coefficients. Alternatively, similar decorrelation effects can be obtained by Linear Prediction method or KL transform [Gal99].

Most statistical methods for ASR assume the signal in each frame is *stationary* and model successive frames as independent (conditioned on the words being spoken). However, this is a poor assumption for real speech. One way to incorporate the correlation between neighbouring frames is to compute the first and second derivatives of the MFCCs and append them to the feature vector. Thus the final feature vector for ASR consists of 39 dimensions in total, and a sequence of these feature vectors are passed to

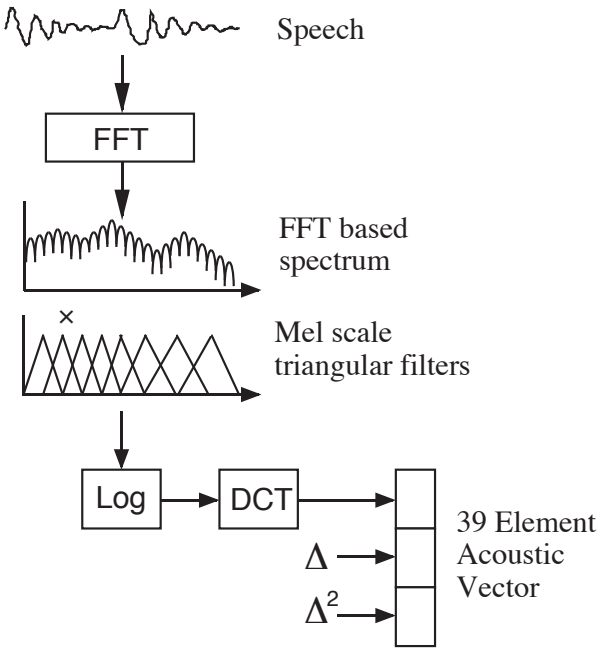

Figure 4.2: An overview of the process of acoustic feature extraction, which is exemplified in MFCC computation [You96].

the recognizer's acoustic model for further processing.

4.3 Acoustic modeling

An acoustic model computes the likelihood $p(\mathbf{X}|\mathbf{W})$ of acoustic feature vectors $\mathbf{X}$ given words $\mathbf{W}$. Most commonly this is done with a hidden Markov model (HMM), which defines a joint distribution over sequences of hidden states (e.g. words) and observations (e.g. feature vectors). An HMM is parameterized by an initial state distribution, a transition matrix, and an emission density in each state [Rab89]. The emission densities in ASR are usually parameterized by Gaussian mixture models (GMMs).

Most recognizers use phonemes to represent the basic units of speech [GY08]. Each phoneme is typically modeled by a left-to-right HMM with three hidden states. A *word* HMM is constructed by concatenating the corresponding HMMs for its constituent phonemes, and the HMM for a sequence of words is constructed similarly by concatenating the word HMMs. Fig. 4.3 illustrates the state space of a 3-state left-to-right

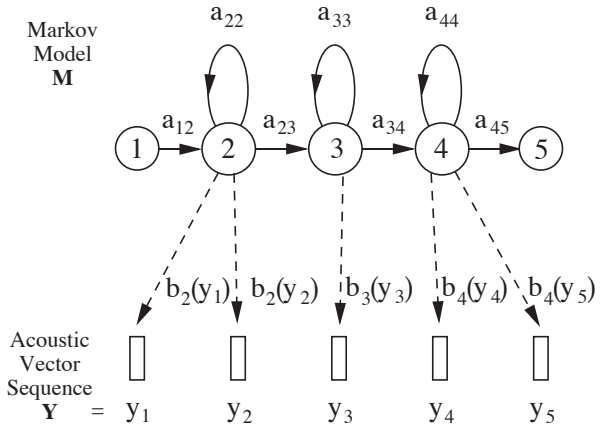

Figure 4.3: A typical 3-state HMM for a phoneme, with one additional state at the front and at the end for silence and transition between phonemes [You96].

HMM, with two additional states at the front and end to model silence and phoneme transitions.

Parameter estimation in HMMs is traditionally handled by the expectation-maximization (EM) algorithm for maximum likelihood estimation:

$$\theta^{MLE} = \text{argmax}_\theta \sum_i \log p(\mathbf{X}_i|\mathbf{W}_i;\theta), \tag{4.3}$$

where θ denotes the parameters to configure the HMM. The HMMs for correct word sequence can be constructed by concatenation. However, when the exact locations of phoneme boundaries are unknown, the states s_t of the HMMs must be treated as hidden variables. An auxiliary function for the log-likelihood $\log p(\mathbf{X}_i, \mathbf{s}; \hat{\theta})$ is given by

$$Q(\hat{\theta}, \theta) = \sum_i \sum_{\mathbf{s}} p(\mathbf{X}_i, \mathbf{s}; \theta) \log p(\mathbf{X}_i, \mathbf{s}; \hat{\theta}), \tag{4.4}$$

where $\hat{\theta}$ denotes the parameters to be updated given the current parameters θ. The updates are obtained by maximizing the right hand side of eq. (4.4), which in turn computes statistics of the posterior distributions $p(s_t|X_t^i;\theta)$. These statistics can be computed efficiently by forward-backward algorithms, and when the emission densities are Gaussian mixture models, closed form expressions can be derived for the maximization and parameter updates [Rab89].

At test times, the search for the most likely word sequence must trace over the whole hypothesis space. This space is usually represented by a trellis, where each node in the trellis is a word (or word sequence) allowed by the language model. The decoding can be efficiently performed by dynamic programming algorithms (e.g. the Viterbi algorithm), with language models helping to eliminate unlikely paths and speed up the search.

4.4 Discriminative Training

As mentioned in the previous section, parameter estimation in generative models is most commonly handled by EM algorithm for maximum likelihood estimation (MLE). The success of MLE is generally based on several assumptions: 1) the underlying parametric model is correct, 2) the true language model is known, and 3) the size of the training data is infinitely large [Nad83]. These assumptions are hard to met for real speech signals, notwithstanding the many efforts that are made in the acoustic feature extraction (i.e., log compression, DCT, and derivative features) to accommodate the framework.

Many have suggested [Vap98] that one should solve classification problems directly rather than estimating a generative model of the data as an intermediate step. Tis observation has raised interest in discriminative training methods, which attempt to minimize the classification error directly instead of estimating the parameters of a generative model for the underlying distribution.

Discriminative training methods have significantly improve the accuracy of ASR in the last two decades. Here we briefly review MLE and compare it with the two most popular discriminative training methods for ASR. For simplicity, we begin by reviewing the basic ideas in the context of multiway classification for i.i.d. data (see Chapter 2). Finally, we discuss the extension to sequential classification, focusing on the challenges that arise in ASR.

4.4.1 Maximum Likelihood Estimation

The decision rule for classifying an unknown data $\mathbf{x}$ into one of the $\mathcal{C}$ classes is

$$\hat{y} = \text{argmax}_c p(c|\mathbf{x}) \propto \text{argmax}_c p(\mathbf{x}|c)p(c). \tag{4.5}$$

If we assume that the prior distribution for each class c is known, then what we need to learn for this decision rule is the class conditional distribution $p(\mathbf{x}|c)$ over $\mathbf{x}$. MLE is the simplest approach to learn the distribution $p(\mathbf{x}|c)$ from training examples; the parameters are estimated by maximizing the likelihood:

$$\theta^{\text{MLE}} = \text{argmax}_\theta \prod_{i=1}^{N} p(\mathbf{x}_i|y_i;\theta) = \text{argmax}_\theta \sum_{i=1}^{N} \log p(\mathbf{x}_i|y_i;\theta), \tag{4.6}$$

where θ is the parameter set and N is the number of training examples. Note that the parameter set θ can be separated into $\mathcal{C}$ disjoint sets $\theta = \{\theta_c\}_{c=1}^{\mathcal{C}}$, and that in MLE, the parameters of each class can be learned *independently*.

The success of MLE is not guaranteed when the data's underlying distribution does not match the form of the model. Nor does it yield the best-performing classifier when the amount of training data is limited. Both these difficulties arise in ASR. Not surprisingly, these limitations of MLE have generated interest in other methods for parameter estimation.

4.4.2 Conditional Maximum Likelihood

The decision rule in eq. (4.5), based on *maximum a posterior* (MAP), suggests another approach to parameter estimation [Nad83]. Instead of maximizing the joint likelihood of training data given the class labels, the intuition behind conditional maximum likelihood (CML) is to estimate parameters which directly maximize the log-probability of correct classification:

$$\begin{aligned} \theta^{\text{CML}} &= \text{argmax}_\theta \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i;\theta) \\ &= \text{argmax}_\theta \sum_{i=1}^{N} \log \frac{p(\mathbf{x}_i, y_i;\theta)}{p(\mathbf{x}_i;\theta)}, \end{aligned} \tag{4.7}$$

We can rewrite the right hand side of eq. (4.7) as the difference between the likelihood of the data given the correct class label and the overall likelihood:

$$\theta^{\text{CML}} = \text{argmax}_\theta \sum_{i=1}^{N} \{\log p(\mathbf{x}_i, y_i; \theta) - \log \sum_c p(\mathbf{x}_i, c; \theta)\}. \tag{4.8}$$

Thus, to maximize this objective function is to increase the gap in log-likelihood between the correct labeling and all other hypotheses.

A related estimator is based on maximizing the mutual information between examples and labels:

$$\theta^{\text{MMI}} = \text{argmax}_\theta \sum_{i=1}^{N} \log \frac{p(\mathbf{x}_i, y_i)}{p(\mathbf{x}_i)p(y_i)}, \tag{4.9}$$

where we omit θ for simplicity. When the prior probabilities of class labels are fixed, the so-called maximum mutual information (MMI) estimator [BBdSM86] in eq. (4.9) is equivalent to the CML estimator in eq. (4.7). This equivalence usually holds in the training of large vocabulary speech recognizers.

4.4.3 Minimum Classification Error

Another appealing approach is minimum classification error (MCE) [JK92], which aims at minimizing the generalization error, i.e. the probability of misclassifying an unlabeled example $\Pr_{(x,y)\sim\mathcal{D}}[g(\mathbf{x}) \neq y]$, where $g(\mathbf{X})$ is the decision function such as the one defined in eq. (4.5), drawn from the same underlying distribution as the training data. Unlike CML, MCE works directly on the misclassification error rate, but both these approaches share the same benefits of discriminative training [NJ02].

MCE algorithm are derived by defining a discriminant function and minimizing an objective function that serves as a surrogate for the misclassification rate. For the discriminant function $g(\mathbf{x})$, we can use the posterior probability

$$g(\mathbf{x}, y) = p(y|\mathbf{x}; \theta); \tag{4.10}$$

thus the Bayes decision rule can be written

$$\hat{y} = \text{argmax}_c g(\mathbf{x}, c). \tag{4.11}$$

The goal of MCE is to directly minimize the number of misclassifications; to do so, we would estimate

$$\theta^{\text{MCE}} = \text{argmin}_\theta \sum_{i=1}^{N} \delta(\hat{y}_i \neq y_i), \tag{4.12}$$

where $\delta(z)$ is an indicator function equal to one if and only if the argument z is true. However, the misclassification count in eq. (4.12) is nondifferentiable and hence difficult to optimize. Juang et al. [JK92] suggested an alternative approach. They defined the surrogate measure:

$$d(\mathbf{x}_n) = -g(\mathbf{x}_n, y_n) + \left[\frac{1}{\mathcal{C}-1} \sum_{c \neq y_n} g(\mathbf{x}_n, c)^\eta \right]^{1/\eta}, \tag{4.13}$$

where $\eta > 0$ is a trainable parameter. Note that as $\eta \to \infty$, eq. (4.13) reduces to

$$\lim_{n \to \infty} d(\mathbf{x}_n) = -g(\mathbf{x}_n, y_n) + g(\mathbf{x}_n, y^*), \tag{4.14}$$

where y^* is the predicted class with the largest discriminant value among classes other than y_n.

The surrogate measure is differentiable, but it does not closely track the misclassification count in eq. (4.12). A final objective function for MCE training is obtained by projecting eq. (4.14) into the interval $[0, 1]$ by sigmoid transfer function:

$$\ell(d(\mathbf{x}_n)) = \frac{1}{1 + \exp(-\alpha d(\mathbf{x}_n) + \beta)}. \tag{4.15}$$

Here α is a heuristically chosen parameter that tunes the sharpness of the sigmoid function. Bounded between 0 and 1, the loss in eq. (4.15) serves as a differentiable approximation of the misclassification count in eq. (4.12). The optimization for MCE training is usually done by gradient-based methods [LM05, MHR+07].

4.4.4 Challenges for sequential classification

New challenges arise when discriminative training is extended to problems in sequential classification. Given training sequences $\{(\mathbf{X}_i, \mathbf{W}_i)\}_{i=1}^N$, where $\mathbf{X}_i$ and $\mathbf{W}_i$ are sequences of observations and class labels, the goal of learning is to estimate parameters that correctly label new observation sequences drawn from the same underlying

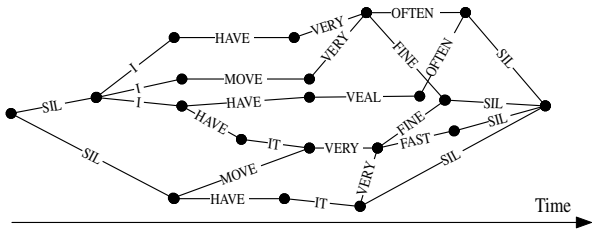

Figure 4.4: An example of a typical word lattice to represent hypotheses for a sentence [GY08].

distribution. In speech recognition, the class labels can be words or phonemes, and for concreteness we assume they are words. Consider the case of MMI as discussed in the previous section. In this case, the parameters are estimated as:

$$\theta^{\text{MMI}} = \text{argmax}_\theta \sum_{i=1}^{N} \log \frac{p(\mathbf{X}_i, \mathbf{W}_i; \theta)}{\sum_{\mathbf{W}} p(\mathbf{X}_i, \mathbf{W}; \theta)}, \tag{4.16}$$

where the denominator in the logarithm sums over all possible word sequences. Note that the number of possible hypotheses grows exponentially with the length of each sequence.

The optimization of the MMI criterion in eq. (4.16) requires maximizing the numerator while minimizing the denominator. The numerator is the only of these terms that appears in ML estimation. The overall computation for MMI training is dominated by the denominator and affected greatly by the size of the vocabulary, the language model, and the contextual acoustic models. For large vocabulary speech recognition, where it is common to use cross-word context-dependent acoustic models and long-span language models, it is generally impossible to compute the sum over all hypotheses in the denominator. To improve the efficiency, most state-of-the-art systems implement a word lattice (Fig. 4.4) that provides a compact representation of the most likely hypotheses [GY08].

For sequential classification, another difficulty of discriminative training arises from the numerical optimization of eq. (4.16). ML estimation is very appealing in ASR due to its simplicity and efficiency; the EM algorithm scales very well , even to large-vocabulary ASR. Early approaches to discriminative training were based on gradient methods which converge more slowly than EM updates for ML estimation. Later, [GKNN91, VOWY97] proposed the extended Baum-Welch (EBW) algorithm for

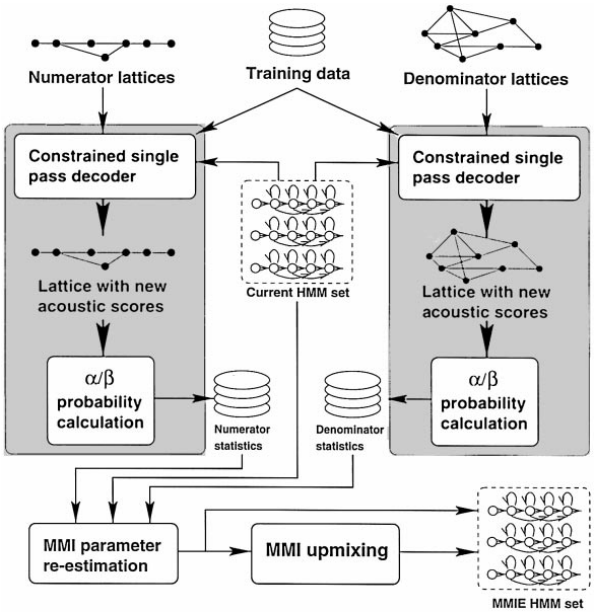

Figure 4.5: A sample framework of discriminative training for large-vocabulary speech systems [VOWY97].

discriminative training. The EBW updates are re-estimation formulae based on statistics that can be computed from forward-backward algorithms. The re-estimation is stabilized by a scaling factor that must be heuristically determined. Currently EBW is the most effective and efficient algorithm for discriminative training in ASR. In large vocabulary speech systems, the algorithm requires the generation of word lattices, and one forward-backward pass for each EBW update and re-scoring of the lattices (see Fig. 4.5). In addition, it is usually implemented in parallel on a cluster of fast machines.

Chapter 5

Baseline Experiments

In Chapter 4 we discussed the general framework for ASR used in modern commercial and laboratory systems. In this chapter, we shift focus to the problem of phoneme recognition, which can be viewed as speech recognition without the benefit of a language model. The experimental settings in phoneme recognition are slightly different than those of word recognition, as described in the previous chapter. Phoneme recognizers have a much smaller hidden state space than word recognizers, making it possible to experiment more easily with alternative learning strategies. For this reason, research in phoneme recognition remains useful for improving our understanding of more general problems in ASR [SRP09, WEK+09, DRrMH10, Cra10, SNE+10].

We begin by reviewing basic notation for continuous-density HMMs (abbreviating to CD-HMMs) and their use for phoneme recognition. We also report several previous benchmarks for acoustic modeling, against which subsequent models will be judged. Unless specified otherwise, the notations here will be used in the remaining chapters of this thesis.

5.1 Continuous-density hidden Markov models

CD-HMMs define a joint probability distribution over sequences of hidden states $\mathbf{s} = \{s_1, s_2, \ldots, s_T\}$ and observations $\mathbf{x} = \{x_1, x_2, \ldots x_T\}$. The joint distribution is expressed in terms of the initial state distribution $\mathcal{P}(s_1)$, the hidden state transition matrix $\mathcal{P}(s_{t+1}|s_t)$, and the emission densities $\mathcal{P}(x_t|s_t)$. In terms of these quantities, the joint

distribution is given by:

$$\mathcal{P}(\mathbf{s}, \mathbf{x}) = \mathcal{P}(s_1) \prod_{t=1}^{T-1} \mathcal{P}(s_{t+1}|s_t) \prod_{t=1}^{T} \mathcal{P}(x_t|s_t). \tag{5.1}$$

For phoneme recognition, each hidden state represents a sub-word linguistic unit (such as a phoneme), and each observation corresponds to an acoustic feature vector. The emission densities for ASR are parameterized by Gaussian mixture models (GMMs), with mixture weights $\mathcal{P}(c|s)$, mean vectors μ_{sc}, and covariance matrices Σ_{sc} for the c^{th} component of each hidden state. In terms of these parameters, the emission densities are given by:

$$\mathcal{P}(x|s) = \sum_c \frac{\mathcal{P}(c|s)}{\sqrt{(2\pi)^d |\Sigma_{sc}|}} e^{-\frac{1}{2}(x-\mu_{sc})^\top \Sigma_{sc}^{-1}(x-\mu_{sc})}. \tag{5.2}$$

Given a sequence of observations $\mathbf{x}$, we can infer the most likely hidden state sequence $\mathbf{s}^*$ as:

$$\mathbf{s}^* = \text{argmax}_s \log \mathcal{P}(\mathbf{s}|\mathbf{x}, \Theta). \tag{5.3}$$

The inference in eq. (5.3) depends on the parameters of the CD-HMM, which we collectively denote by Θ. The right hand side of eq. (5.3) can be computed efficiently by dynamic programming. In particular, of all possible sequences of hidden states, the Viterbi algorithm recursively constructs the one with the highest posterior probability.

5.2 Generative versus discriminative approaches

The simplest form of training for CD-HMMs is ML estimation. ML estimation is based on viewing CD-HMMs as a generative model of speech. For joint examples $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ of observation sequences and target state sequences, this approach aims to maximize the joint log-likelihood:

$$\Theta_{\text{ML}} = \text{argmax}_\Theta \sum_{n=1}^{N} \log \mathcal{P}(\mathbf{y}_n, \mathbf{x}_n|\Theta). \tag{5.4}$$

Once the overall model architecture is specified, maximum-likelihood estimates of the parameters in CD-HMMs may be computed by the Expectation-Maximization (EM) algorithm. The EM algorithm monotonically increases the log-likelihood in eq. (5.4)

with each update, and does not involve tuning parameters such as learning rates. All these properties make it very attractive as a starting point for ASR.

However, ML estimation has one serious drawback: increasing the joint likelihood in eq. (5.4) does not generally decrease the error rate of the recognizer. Recent empirical [NJ02] and theoretical [LJ08] analyses have highlighted the shortcomings of ML estimation when the estimated models are not perfectly matched to the data. To overcome these shortcomings, we must consider discriminative methods for parameter estimation in CD-HMMs.

Discriminative training of CD-HMMs has a long history in ASR [BBdSM86, Nad83, JK92], and new work continues to appear in this area. The fundamental idea behind discriminative training is to seek parameters that minimize the error rate rather than attempting to model the data itself. One popular method for discriminative training in ASR is based on maximizing the mutual information (MMI) between observations and states:

$$\Theta_{\mathrm{MMI}} = \mathrm{argmax}_{\Theta} \sum_{n=1}^{N} \log \frac{\mathcal{P}(\mathbf{x}_n, \mathbf{y}_n|\Theta)}{\mathcal{P}(\mathbf{x}_n|\Theta)\mathcal{P}(\mathbf{y}_n|\Theta)}. \tag{5.5}$$

The maximization is typically done by gradient ascent or extended Baum-Welch updates; both methods require computing derivatives of the right hand side with respect to the parameter Θ. Closely related to MMI is conditional maximum likelihood (CML) estimation:

$$\Theta_{\mathrm{CML}} = \mathrm{argmax}_{\Theta} \sum_{n=1}^{N} \log \mathcal{P}(\mathbf{y}_n|\mathbf{x}_n, \Theta). \tag{5.6}$$

CML differs only slightly from MMI in its estimation of transition probabilities and language model parameters; the methods are equivalent if these parameters are not updated. Another popular discriminative method is minimum classification error (MCE), which directly minimizes the number of sequence misclassifications. In MCE, the parameters are found by optimizing:

$$\Theta_{\mathrm{MCE}} = \mathrm{argmin}_{\Theta} \sum_{n=1}^{N} \mathrm{sign} \left[\max_{s \neq y_n} \log \frac{\mathcal{P}(\mathbf{x}_n, \mathbf{s}|\Theta)}{\mathcal{P}(\mathbf{x}_n, \mathbf{y}_n|\Theta)} \right], \tag{5.7}$$

where $\mathrm{sign}[z] = 1$ if $z > 0$ and $\mathrm{sign}[z] = 0$ if $z \leq 0$. Since the right hand side of eq. (5.7) is nondifferentiable, MCE usually replaces the *max* function by a *softmax* function, then optimizes the parameters by gradient ascent.

Discriminative training of CD-HMMs is more complicated than ML estimation for several reasons: (i) probabilities (and their gradients) must be computed not only for desired state sequences, but also for competing ones; (ii) many update rules involve learning rates, which must be carefully tuned; (iii) convergence is not as fast or as stable as the EM algorithm for ML estimation.

5.3 TIMIT corpus

All our experiments in phoneme recognition are benchmarked on the TIMIT speech corpus [LKS86, ZSG90]. The waveforms in TIMIT were digitally recorded at 20kHz 16-bit PCM format in a quiet environment, then downsampled to 16kHz for further processing. Each wavefile in the TIMIT corpus is associated with an orthographic transcription (*.txt*), a time-aligned word transcription (*.wrd*), and a time-aligned phonetic transcription (*.phn*).

TIMIT was jointly designed by researchers at MIT, SRI International, and Texas Instruments (TI). The database contains 2342 distinct sentences and 630 speakers. The sentences are phonetically balanced to represent the full diversity of pronunciations in American English. The sentences can be categorized into three types:

- *SA* sentences: calibration sentences provided by SRI which illustrate the difference between dialects. The two SA sentences are the same across all dialects in the corpus.

- *SX* sentences: phonetically compact sentences provided by MIT which cover all practical phonetic pairs. A total of 450 distinct SX sentences are evenly distributed throughout the corpus.

- *SI* sentences: randomly selected sentences provided by TI which illustrate the same phonetic sequence in different contexts. Every SI sentence is unique, and each speaker in the corpus utters three of them.

Each speaker in the TIMIT corpus reads two SA sentences, five SX sentences, and three SI sentences; in total there are 630 speakers and 6300 sentences. The speakers (70% are male and 30% are female) represent eight dialectal regions of the United States. In our

Table 5.1: The list of phonemes used in TIMIT corpus. The corresponding pronunciation is underlined in the example .

Phoneme	Example	Phoneme	Example
epi	*silence*	K	kit
pau	*start/end*	DX	batter
IY	beat	JH	judge
IH	bit	CH	church
EH	bet	S	sat
EY	bait	SH	shut
AE	bat	Z	zoo
AA	Bob	ZH	azure
AW	down	F	fat
AY	buy	TH	thing
AH	but	V	vat
AO	bought	DH	that
OY	boy	M	met
OW	boat	N	net
UH	book	NG	sing
UW	boot	EN	button
ER	bird	L	let
AX	about	R	rent
IX	roses	W	wit
B	bet	Y	you
D	debt	HH	hat
G	get	EL	battle
P	pet	BCL	*glottal stop*
T	ten	QCL	*glottal stop*

experiments, we omit the SA sentences because they are the same across the speakers and might bias the training.

The corpus is partitioned into a non-overlapping training set and test set. The training set contains 462 speakers, with 8 sentences uttered by each speaker, which results in 3696 sentences in total. The complete test set has a subset known as the "core" test set. We evaluate our models on the *core* test set, which contains 192 sentences from 24 speakers. We validate our models on a separate development set, also selected from the complete test set, which contains 400 sentences from 50 speakers.

The lexicon in TIMIT contains 48 phonemes commonly used in American En-

glish. Table 5.1 shows each phoneme in ARPABET notation along with an example [RJ93].

5.4 Existing benchmarks

The benefits of discriminative training have been demonstrated in many different tasks and applications. We used the TIMIT speech corpus [LKS86] to evaluate the competing models discussed in this thesis. The speech signals in this corpus have been manually segmented and aligned with their phonetic transcriptions. These transcriptions provide ground-truth labels for benchmarking different acoustic models on the problem of phoneme recognition.

We adopted the same methodology as recent benchmarks on this data set [SS09]. For the front end, we computed acoustic feature vectors of mel-frequency cepstral coefficients on sliding windows of speech, with 39 dimensions in total. For each utterance, we performed cepstral mean subtraction, but not endpointing. For the back end, we trained CD-HMMs using the manually aligned phonetic transcriptions as target hidden states. We did not introduce or optimize word insertion probabilities to lower the frame and phone error rates. The CD-HMMs had 48 hidden states (one per context-independent phoneme) and GMMs that varied in size from one to 128 mixture components. We used the standard partition of the TIMIT corpus, yielding roughly 1.1 million, 120K, and 57K frames respectively for training, test, and holdout data. This standard partition corresponds to 5 hours of training utterances and 30 minutes of test utterances.

We measured performance by comparing the hidden state sequences inferred by Viterbi decoding of CD-HMMs to the phonetic transcriptions provided by the TIMIT corpus. In calculating error rates, we followed the standard practice of mapping 48 phonetic classes down to 39 broader categories [LH88]. In general, we report two types of errors: the frame error rate (FER), computed simply as the percentage of misclassified frames, and the phone error rate (PER), computed from the edit distances between ground truth and Viterbi decodings [LH88]. The phone error rate provides the more relevant metric for ASR. However, in some instances, we also report the frame error rate because it is more directly minimized by the algorithms we study in later sections.

Table 5.2: Phone error rates for CD-HMMs of varying size on the TIMIT speech corpus, as obtained by maximum likelihood (ML), conditional maximum likelihood (CML), and minimum classification error (MCE) estimation. The results in the first four rows are from previous benchmarks [SS09]. The left column shows the number of mixture components per GMM.

# mixture component	Phone Error Rate (%)		
	ML	CML	MCE
1	41.5	36.4	35.6
2	38.0	34.6	34.5
4	34.9	32.8	32.4
8	32.3	31.5	30.9
16	30.8		
32	31.8		
64	33.4		
128	35.9		

Table 5.2 presents previous benchmarks [SS09] on the TIMIT speech corpus, as well as some additional results on larger models trained by ML. The table shows the phone error rates of CD-HMMs trained by ML, CML, and MCE. Note that discriminative training leads to significantly lower error rates than ML estimation for models of the same size. The results in Table 5.2 will provide useful baselines for the models we discuss in subsequent sections. Also, except where otherwise noted, the ML models in Table 5.2 were used to initialize all discriminatively trained models mentioned in this thesis.

Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 6

Online Updates for HMMs

This chapter describes our framework for online learning of discriminatively trained HMMs. Discriminative training was originally formulated for classification of i.i.d data [Efr75, Nad83]. After years of effort, researchers in ASR and NLP have extended the idea to sequential data [Mer88, KVY93, WP00]. One important research in this line of work was [Col02], which applied Perceptron-style online learning to discriminative training of discrete HMMs. Motivated by the results of [Col02], we consider a similar scheme for discriminative training of continuous-density HMMs - particularly the sort used for acoustic modeling in ASR. The continuous-density functions in these HMMs are parameterized by Gaussian mixture models (GMMs). Conventionally, these GMMs are parameterized in terms of their means, covariance matrices and mixing weights. We show how to reparameterize the GMMs for more efficient online learning by Perceptron-style updates.

This chapter is organized as follows. In section 6.2, we describe the perceptron algorithm for HMMs in general terms. In sections 6.3 through 6.5, we consider how to parameterize the acoustic models used in ASR for this type of training and discuss various issues that arise from different parameterizations. Finally, in section 6.6, we present experimental results on the TIMIT speech corpus. Our results highlight the parameterizations of acoustic models that yield the most consistent and rapid reductions in phone error rates.

6.1 Introduction

Discriminative training of CD-HMMs has a long history in ASR [BBdSM86, Nad83, JK92], and new work continues to appear in this area. The fundamental idea behind discriminative training is to seek parameters that explicitly minimize the error rate rather than attempting to model the data itself. Discriminative training of CD-HMMs is more complicated than ML estimation for several reasons: (i) log-likelihoods must be computed not only for desired state sequences, but also for competing ones; (ii) most update rules involve some form of gradient descent, requiring careful tuning of learning rates; (iii) convergence is not generally as fast as the EM algorithm for ML estimation. Therefore, it is useful to consider procedures that simplify or accelerate discriminative training. In this chapter, we explore the potential of online updates to achieve these goals.

Our approach is motivated by one of the simplest and oldest algorithms for online learning: namely, the perceptron [Ros58]. Perceptrons use a mistake-driven update rule to learn linear decision boundaries between classes of positively and negatively labeled examples (see Chapter 3). An exciting line of recent work has generalized the perceptron algorithm to discriminative training of discrete HMMs [Col02]. The perceptron algorithm for discrete HMMs combines simple additive updates with Viterbi decoding of training examples. On problems in part-of-speech tagging [TKMS03] and base noun phrase chunking [KM01], this algorithm outperformed other leading approaches to discriminative training. We seek to replicate these successes in HMMs for ASR.

New difficulties arise in the perceptron-like training of CD-HMMs that are not present in discrete HMMs. These difficulties are rooted in the parameterization of emission densities. For example, in CD-HMMs, online updates must adapt the means and covariance matrices of multivariate Gaussian distributions. However, simple, additive updates can violate the positive definiteness of covariance matrices, thus requiring further computation to maintain these constraints.

6.2 Mistake-driven updates

Perceptron training in HMMs is based on a so-called discriminant function over observation and hidden state sequences:

$$\mathcal{D}(\mathbf{x}, \mathbf{s}) = \log \mathcal{P}(s_1) + \sum_{t>1} \log \mathcal{P}(s_t|s_{t-1}) + \sum_t \log \mathcal{P}(x_t|s_t). \tag{6.1}$$

The discriminant function is essentially the logarithm of the joint probability distribution in eq. (5.1). For simplicity, we have suppressed the dependence of the discriminant function on the parameters Θ of the CD-HMM. In terms of the discriminant function, a target state sequence $\mathbf{y}$ will be correctly inferred from the observation sequence $\mathbf{x}$ if:

$$\forall \mathbf{s} \neq \mathbf{y}, \quad \mathcal{D}(\mathbf{x}, \mathbf{y}) > \mathcal{D}(\mathbf{x}, \mathbf{s}). \tag{6.2}$$

Note that eq. (6.2) defines a set of inequalities for all incorrect transcriptions $\mathbf{s} \neq \mathbf{y}$. In our experiments, the target state sequences are the manually aligned phonetic transcriptions in the TIMIT corpus. In this work, we make the simplifying assumption that there is a unique target state sequence (as opposed to considering multiple valid target sequences that decode to the same word sequence).

In general, it is not possible for a model to satisfy all the constraints in eq. (6.2). We use the following loss function [SS07b] to measure the total constraint violation across the entire training corpus:

$$\mathcal{L}(\Theta) \; = \; \sum_n \left[\max_{\mathbf{s} \neq \mathbf{y}_n} \mathcal{D}(\mathbf{x}_n, \mathbf{s}) - \mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) \right]^+, \tag{6.3}$$

where $[z]^+ = \max(z, 0)$ indicates the nonnegative hinge function. The right hand side of eq. (6.3) computes a weighted count of the training utterances that do not satisfy the constraints in eq. (6.2). In particular, each incorrectly decoded utterance is weighted by the log-likelihood gap between the correct transcription and the Viterbi decoding, as computed by eq. (5.3).

To minimize the loss function in eq. (6.3), we consider the online, mistake-driven update:

$$\Theta \leftarrow \Theta + \eta \frac{\partial}{\partial \Theta} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) \right], \tag{6.4}$$

where $\eta > 0$ is a carefully chosen learning rate. Note that the update in eq. (6.4) is only applied when Viterbi decoding returns an incorrect transcription $\mathbf{s}_n^* \neq \mathbf{y}_n$. The update can be viewed as a form of stochastic gradient descent [Bot04] on the loss function in eq. (6.3), which has also been studied in the related context of graph transformer networks [BLB97]. The gradient in eq. (6.4) computes the fastest search direction in parameter space to minimize the log-likelihood gap between target and inferred state sequences. For discriminative training, we may also adapt the parameters of acoustic models in such a way that they no longer define a properly normalized joint distribution. In particular, we need not enforce sum-to-one constraints on the rows of the transition matrix nor the mixture weights of GMMs.

Perceptron training updates parameters in a sequential manner, looping through all the training examples until either the algorithm converges or no longer reduces the average number of classification errors. We follow a similar procedure for updating the parameters of acoustic models for ASR. In general, mistake-driven updates will not converge to a fixed set of parameter estimates if the training examples cannot be perfectly classified. However, convergence to a fixed set of parameter estimates can be obtained by averaging the parameters from perceptron training across different updates of the training examples [FS99, Gen02]. In practice, this sort of averaging reduces the noise in the parameter vector by damping fluctuations in the decision boundary that occur during training. Let $\Theta^{(j)}$ represent the parameter estimates after the perceptron update in eq. (6.4) has been applied for the j^{th} time. We compute the averaged parameters $\tilde{\Theta}^{(r)}$ after r updates as:

$$\tilde{\Theta}^{(r)} = \frac{1}{r} \sum_{j=1}^{r} \Theta^{(j)}. \tag{6.5}$$

Note that these averaged estimates are not themselves used during training; they are only computed after training, then used for the classification of new test examples. In addition to parameter averaging, convergence may also be obtained by decreasing the learning rate over time; however, experimenting with this strategy, we found it to be much less effective than parameter-averaging.

6.3 Parameterization of GMMs

Conventionally, CD-HMMs are parameterized in terms of transition matrices and emission densities. The choice of parameterization plays an important role in online learning. For example, consider the update rules for the mixture weights $\mathcal{P}(c|s)$ and the diagonal elements of the covariance matrices Σ_{sc}. Simple additive updates to these parameters may not preserve their nonnegativity, which is necessary for the discriminant function in eq. (6.1) to be well-defined for all possible observation and state sequences. More generally, the choice of parameterization can significantly affect the rate of convergence of online learning, as well as the nature of the averaging in eq. (6.5).

In the rest of this section, we flesh out these issues, concentrating mainly on the parameterization of the GMMs. In general, the GMM parameters in HMMs play a much more important role than the transition probabilities; moreover, the latter are easily over-trained. Thus, in practice, if the transition probabilities are updated at all by discriminative training, they should be adapted by a very small learning rate. We did not update the transition probabilities in our experiments.

The GMMs in CD-HMMs are conventionally parameterized in terms of the mixture weights $\mathcal{P}(c|s)$, means μ_{sc}, and covariance matrices Σ_{sc} associated with different hidden states and mixture components. However, the most straightforward online updates are given in terms of the log-mixture weights $\nu_{sc} = \log \mathcal{P}(c|s)$ and inverse covariance matrices Σ_{sc}^{-1}. The mixture weights are best updated in the log domain to ensure that they remain nonnegative. It is also simpler to compute the derivatives in the update rule, eq. (6.4), with respect to the inverse covariance matrices Σ_{sc}^{-1} than the covariance matrices Σ_{sc}. For the GMM parameters in CD-HMM, these considerations lead to online updates of the form:

$$\nu_{sc} \leftarrow \nu_{sc} + \eta \frac{\partial}{\partial \nu_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) \right], \tag{6.6}$$

$$\mu_{sc} \leftarrow \mu_{sc} + \eta \frac{\partial}{\partial \mu_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) \right], \tag{6.7}$$

$$\Sigma_{sc}^{-1} \leftarrow \Sigma_{sc}^{-1} + \eta \frac{\partial}{\partial \Sigma_{sc}^{-1}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) \right]. \tag{6.8}$$

The last update in eq. (6.8) can violate the constraint that the inverse covariance matrix Σ_{sc}^{-1} must be positive definite; when this happens, the zero or negative eigenvalues of

the updated matrix must be thresholded to some small positive value so that individual Gaussian distributions, computed from eq. (5.2), remain normalizable and finite.

Though simple in concept, the stochastic gradient descent in eqs. (6.6–6.8) may require the careful tuning of multiple learning rates in order to succeed. Alternatively, a common strategy is to only optimize the mean parameters of GMMs.

6.4 Reparameterization of GMMs

Building on ideas from previous work [SS09], we investigate a reparameterization of GMMs that aggregates the mixture weight, mean, and covariance matrix parameters associated with each Gaussian mixture component into a single augmented matrix. Let

$$\gamma_{sc} = -\log\left(\frac{\mathcal{P}(c|s)}{\sqrt{(2\pi)^d|\Sigma_{sc}|}}\right) \tag{6.9}$$

denote the log of the scalar prefactor that weights each Gaussian mixture component. Then for each Gaussian mixture component, consider the matrix:

$$\Phi_{sc} = \begin{bmatrix} \Sigma_{sc}^{-1} & -\Sigma_{sc}^{-1}\mu_{sc} \\ -\mu_{sc}^\top\Sigma_{sc}^{-1} & \mu_{sc}^\top\Sigma_{sc}^{-1}\mu_{sc} + \gamma_{sc} \end{bmatrix}. \tag{6.10}$$

In eq. (6.10), the upper left block of the matrix Φ_{sc} is simply the inverse covariance matrix Σ_{sc}^{-1}, while the other elements of Φ_{sc} are determined by the interaction of the mean μ_{sc} and covariance matrix Σ_{sc}. Note that in terms of this matrix, we can rewrite eq. (5.2) as:

$$\mathcal{P}(x|s) = \sum_c e^{-\frac{1}{2}z^\top\Phi_{sc}z} \quad \text{where} \quad z = \begin{bmatrix} x \\ 1 \end{bmatrix}. \tag{6.11}$$

We can use the reparameterization in eqs. (6.10–6.11) to adapt the matrices Φ_{sc} by mistake-driven updates, as opposed to the GMM parameters in the previous section. In this way, we can replace the three separate updates in eqs. (6.6–6.8) by the single update:

$$\Phi_{sc} \leftarrow \Phi_{sc} + \eta\frac{\partial}{\partial\Phi_{sc}}\left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right]. \tag{6.12}$$

Our experiments in section 6.6 compare the performance of this single update to the separate updates in eqs. (6.6–6.8).

We impose a further constraint on the matrices Φ_{sc} that is not immediately implied by the reparameterization in eq. (6.10); namely, we require them to be positive semidefinite. Although the inverse covariance matrices Σ_{sc}^{-1} are constrained to be positive definite, the matrices Φ_{sc} in eq. (6.10) do not inherit this property if the scalar prefactors γ_{sc} are defined from existing GMMs as in eq. (6.9). Does this constraint limit the representational capacity of our acoustic models? Naively, a positive semidefinite constraint $\Phi_{sc} \succeq \mathbf{0}$ appears to suggest that the probability density $P(x|s)$ in eq. (6.11) cannot be arbitrarily large—that is, it cannot be arbitrarily peaked or concentrated about its mean value.

In fact, the positive semidefinite constraints on Φ_{sc} do not limit the representational capacity of our acoustic models. The capacity is preserved by relaxing the constraint that these models define properly normalized continuous densities over the acoustic feature space. Note that in CD-HMMs, the Viterbi sequences are determined by the likelihood ratios of emission densities in different states. Given any CD-HMMs, with arbitrarily peaked emission densities, consider the unnormalized CD-HMM whose emission densities are multiplied by a constant factor across all states. The likelihood ratios between states are unchanged. However, if the multiplicative factor is sufficiently large, then all the likelihood ratios can be preserved by the reparameterization in eqs. (6.10–6.11), provided that eq. (6.9) incorporates an additive offset from the logarithm of the multiplicative factor. In this way, the reparameterized (unnormalized) acoustic models in this section can be initialized to replicate the exact decoding procedures of any CD-HMM.

Note that like the earlier update in eq. (6.8) for the inverse covariance matrix, the update in eq. (6.12) can violate the constraint that the matrix Φ_{sc} must be positive semidefinite. When this happens, the updated matrix must be projected back onto the cone of positive semidefinite matrices.

Unlike the earlier update in eq. (6.8) for the inverse covariance matrix, we can also allow the matrix Φ_{sc} to have strictly zero eigenvalues. In particular, though eq. (5.2) is not defined for singular covariance matrices, eq. (6.11) is perfectly well defined for all positive semidefinite matrices $\Phi_{sc} \succeq 0$. Thus the online update in eq. (6.12) can learn to use unnormalized Gaussians with unbounded (though nonnegative) variance

if they do indeed lead to fewer classification errors. Essentially, zero eigenvalues in the matrices Φ_{sc} indicate directions (perhaps invariances) in feature space that are not useful for large margin classification. However, we do not allow the matrices Φ_{sc} to have negative eigenvalues; otherwise, observations would be more likely to be associated with particular states even as they deviated further away from the centroids of those states.

We emphasize again that the update in eq. (6.12) effectively removes the constraint that the Gaussian distributions are properly normalized. Note that for a properly normalized Gaussian distribution, the bottom diagonal matrix element of Φ_{sc} in eq. (6.10) is completely determined by the mean μ_{sc} and covariance matrix Σ_{sc}. However, in discriminative training, we can update these matrix elements independently, no longer enforcing normalization constraints on each Gaussian mixture component. The resulting model does not define a proper density over acoustic feature vectors; however, it uses the same decoding procedures (based on dynamic programming) as CD-HMMs.

6.5 Matrix factorizations

The update in eq. (6.12) has the potentially serious drawback that it can violate the constraint that the matrices Φ_{sc} are positive semidefinite. Unlike the constraints of normalizability or bounded variance that were relaxed in the last section, these constraints are important to enforce: otherwise a particular state s and mixture component c could be deemed more and more likely even as observed acoustic feature vectors deviated further and further away from the state and mixture component's centroid μ_{sc}. Though updated matrices can be projected back into the cone of positive semidefinite matrices whenever these constraints are violated, projected gradient methods tend to converge more slowly than unconstrained methods, particularly when the projection and gradient steps work at cross purposes.

We can reformulate our problem as an unconstrained optimization by a further reparameterization, writing each matrix Φ_{sc} as the product of another matrix Λ_{sc} and its transpose $\Lambda_{sc}^{\top}$. The factorization

$$\Phi_{sc} = \Lambda_{sc}\Lambda_{sc}^{T} \tag{6.13}$$

makes explicit that the matrix Φ_{sc} is positive semidefinite. With this factorization, we

can replace the update in eq. (6.12) by stochastic gradient descent in the matrix Λ_{sc}:

$$\Lambda_{sc} \leftarrow \Lambda_{sc} + \eta \frac{\partial}{\partial \Lambda_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) \right]. \tag{6.14}$$

Note that in this update, the square matrices Λ_{sc} (of the same size as Φ_{sc}) are completely unconstrained.

The update in eq. (6.14) has potential advantages and disadvantages. As a form of unconstrained optimization, it has the potential advantage of faster convergence since it does not involve a projected gradient step. On the other hand, it has the potential disadvantage of creating an optimization landscape with more local minima. In particular, note that for the special case in which each Gaussian mixture model has only one mixture component, the difference of discriminant functions is actually linear in the matrices Φ_{sc}. This simple optimization landscape is lost with the factorization in eq. (6.14): the discriminant function is neither linear nor convex in the matrices Λ_{sc}. Our experiments in section 6.6.2 attempt to determine which potential advantages and disadvantages of this matrix factorization are realized in practice.

We note that the factorization in eq. (6.13) is not unique. While a matrix square root satisfying eq. (6.13) can be computed by singular value decomposition, the matrix Λ_{sc} is not uniquely determined unless additional constraints are imposed. One way to obtain a unique factorization is by constraining Λ_{sc} to be positive semi-definite; however, such a constraint is precisely what we hoped to finesse by factorizing the matrix Φ_{sc} in the first place. Another way to obtain a unique factorization – the Cholesky factorization – is by constraining Λ_{sc} to be a lower triangular matrix. We were curious whether such a factorization would accelerate learning (because a lower triangular matrix has fewer parameters to estimate than a full matrix) or decelerate learning (because optimizations sometimes converge more quickly in an enlarged parameter space). Since the optimization is non-convex, we were also curious whether such a factorization might provide a consistently better initialization. In section 6.6.2, we evaluate and present results for two ways of updating the matrices Λ_{sc}: one that constrains them to be lower triangular, and one that does not.

The factorization in eq. (6.13) raises another issue related to the averaging of parameter estimates as in eq. (6.5). For training, we can update the matrices Φ_{sc} directly by eq. (6.12) or indirectly by eq. (6.14). However, the best approach for training does

not necessarily correspond to the best approach for testing with smoothed parameter estimates. Using the notation of eq. (6.5), one approach is to average the parameter estimates for Φ_{sc} as:

$$\tilde{\Phi}_{sc}^{(r)} = \frac{1}{r}\sum_{j=1}^{r}\Phi_{sc}^{(j)} = \frac{1}{r}\sum_{j=1}^{r}\Lambda_{sc}^{(j)}\Lambda_{sc}^{(j)\top}. \tag{6.15}$$

Another approach is to average the parameter estimates for Λ_{sc}, then to square their average as:

$$\tilde{\Phi}_{sc}^{(r)} = \tilde{\Lambda}_{sc}^{(r)}\tilde{\Lambda}_{sc}^{(r)\top}, \text{ where } \tilde{\Lambda}_{sc}^{(r)} = \frac{1}{r}\sum_{j=1}^{r}\Lambda_{sc}^{(j)}. \tag{6.16}$$

In section 6.6.4, we evaluate and present results for both types of averaging.

6.6 Experiments

We experimented on the TIMIT speech corpus (see section 5.4) to evaluate the online updates described in sections 6.2–6.5. Our experiments were designed not only to assess the potential benefits of discriminative training, but also to compare different mistake-driven updates for online learning of HMMs.

Online updating of acoustic models for ASR raises several issues that do not arise in perceptron training of discrete HMMs. Our experiments addressed three main issues: (i) how should the GMMs be parameterized, in the same way as for ML estimation (section 6.3), or by aggregating the parameters for each mixture component into a single matrix (section 6.4)? (ii) how should we enforce the positive semidefiniteness constraints on matrix parameters, by projected gradient methods in the original parameter space or by reparameterizing the matrices using singular value decompositions or Cholesky factorizations (section 6.5)? (iii) in which parameter space should we average to obtain smoothed parameter estimates for testing (section 6.5)? Our experimental results provide fairly definitive answers to these questions.

Before presenting the results, we briefly discuss our methodology. We examined test error rates across a wide range of model sizes by varying the number of Gaussian mixture components per hidden state. We report these results for acoustic models of different sizes to illustrate various general trends. In practice, however, the correct model

size is not known in advance; it is a hyperparameter that must be determined by the performance on held-out data. Thus, in each table of results that follows, we also indicate in **boldface** the test error rate of the model that had the lowest phone error rate on the TIMIT development set. The model selected in this way was often though not always the best model on the test set.

6.6.1 Overall benefits of online learning

We begin by reporting results that confirm the well-known benefits of discriminative training and online learning. Table 6.1 compares the best-performing CD-HMMs obtained by ML estimation to the best performing acoustic models obtained by online, mistake-driven updates. The latter used the matrix update in eqs. (6.13–6.14) and the averaging scheme in eq. (6.15). The results show that the online updates lead to significant reduction in both frame and phone error rates for models with up to sixteen Gaussian mixture components per hidden state. The improvements in frame error rates are larger than the improvements in phone error rates; this discrepancy reflects the fact that the discriminant function more closely tracks the Hamming distance (not the edit distance) between target and Viterbi phone sequences. For reference, Table 6.1 also shows previously published benchmarks [SS09] from the two most popular batch approaches to discriminative training. It is interesting that for all model sizes, the online updates outperform these batch approaches.

Though training times vary from experiment to experiment, we observed the following general trend. For the smallest models (e.g., 1-2 mixture components per hidden state), the discriminative training took much longer than the initial ML estimation; for medium-sized models (e.g., 4-8 mixture components per hidden state), the discriminative training took roughly the same amount of time; finally, for the largest models (e.g., 16-32 mixture components per hidden state), the discriminative training took less time than the initial ML estimation. It seems that the speed-ups from online learning are most pronounced for large model sizes; in particular, in this regime, the speed-ups from stochastic gradient descent appear to more than offset the extra computations (e.g., Viterbi decoding of training utterances) required for discriminative training.

Table 6.1: Frame and phone error rates for acoustic models of varying size, as obtained by maximum likelihood (ML) estimation, online mistake-driven updates, and popular batch methods for discriminative training. The batch results for conditional maximum likelihood (CML) and minimum classification error (MCE) are reproduced from previous benchmarks [SS09]. The left column shows the number of mixture components per GMM.

#	Frame Error Rate (%)		Phone Error Rate (%)			
mix	ML	online	ML	online	CML	MCE
1	39.7	31.4	41.5	33.6	36.4	35.6
2	36.2	30.1	38.0	32.3	34.6	34.5
4	33.1	29.5	34.9	31.4	32.8	32.4
8	30.7	28.8	32.3	30.1	31.5	30.9
16	29.5	**28.6**	30.8	**29.7**		
32	29.9	29.3	31.8	30.9		

6.6.2 Benefits of reparameterization

As noted in section 6.3, for the conventional parameters of GMMs, it is often necessary to tune separate learning rates for discriminative training to succeed. Our experiments bore out these difficulties. The left column of Table 6.2 shows our best results from discriminative training with the conventional parameterization of GMMs. In fact, these results were obtained by updating just the mixture weights and mean vectors of the GMMs; despite extensive experimentation, we were unable to obtain further improvements by updating the inverse covariance matrices in parallel or even while holding the other parameters fixed. Our results are consistent with previous anecdotal observations in ASR: in practice, most of the performance gains in discriminative training have been realized by optimizing the mean parameters in GMMs.

The reparameterization in section 6.4 greatly simplifies both the form of the discriminant function, eq. (6.11), and the resulting online updates. The results in the rightmost column of Table 6.2 reveal the benefits of this approach. These results were obtained using the reparameterization in eq. (6.10), the update in eq. (6.12), and the averaging in eq. (6.5). Note that mistake-driven updates based on the parameters Φ_{sc} from eq. (6.10) leads to significantly lower frame error rates across all model sizes.

Table 6.2: Frame error rates from discriminative training of acoustic models with different forms of online updates: updating (ν, μ, Σ^{-1}) in eqs. (6.6-6.8) versus updating Φ in eqs. (6.10–6.12).

#	Frame Error Rate (%)	
mix	(ν, μ, Σ^{-1})	Φ
1	37.0	32.2
2	36.5	31.5
4	35.8	31.0
8	33.9	30.9
16	31.8	30.5
32	**30.1**	**30.4**

6.6.3 Benefits of matrix factorization

We also experimented with the matrix factorization in eq. (6.13). Updating the matrices Λ_{sc} by eq. (6.14) led to the results shown in Table 6.3. The middle column shows the results when the matrices Λ_{sc} were unconstrained and initialized by singular value decomposition; the right column shows the results when the matrices Λ_{sc} were constrained to be lower diagonal and initialized by Cholesky factorization. For comparison, the left column repeats the results from Table 6.2 for updating the matrices Φ_{sc} by eq. (6.12). Note how the unconstrained factorization in eq. (6.13) leads to consistent further improvements beyond those obtained by the reparameterization in eq. (6.10). The factorization also leads to much faster convergence as measured by the numbers of sweeps through the training data (shown in parentheses). Finally, as an additional benefit, the factorized update also avoids the extra computation required to project the updated parameters Φ_{sc} back into the space of positive semidefinite matrices.

Fig. 6.1 graphically illustrates the much faster convergence of the online, mistake-driven updates using the matrix factorization in eq. (6.13). The figure compares the frame error rates on the training and validation sets during training for the top left (Φ) and middle (Λ-SVD) results in Table 6.3. When updating the matrices Λ_{sc} using eq. (6.14), the training error drops rapidly, and the acoustic models appear to start overfitting after just a few sweeps through the training data. By contrast, when updating the matrices Φ_{sc} using eq. (6.12), the training and holdout error rates drop much more

Table 6.3: Frame error rates from the update in eq. (6.12) versus the update in eq. (6.14). For the latter, we studied two different forms of matrix factorization, one using singular value decomposition (SVD), one using Cholesky factorization. For each result, the number of sweeps through the training data is shown in parentheses.

# mix	Frame Error Rate (%)					
	Φ		Λ-SVD		Λ-Cholesky	
1	32.2	(243)	31.4	(32)	35.5	(149)
2	31.5	(258)	30.1	(37)	35.6	(61)
4	31.0	(296)	29.5	(6)	32.3	(2)
8	30.9	(131)	28.8	(2)	31.4	(2)
16	30.5	(7)	**28.6**	(3)	**29.0**	(2)
32	**30.4**	(2)	29.3	(3)	29.6	(3)

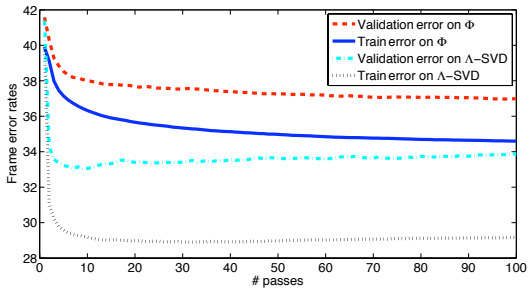

Figure 6.1: Comparison of online, mistake-driven updates with and without the matrix factorization in eq. (6.13). See text for details.

slowly.

6.6.4 Benefits of averaging

Parameter averaging is an effective technique for reducing the fluctuations inherent to online learning. Table 6.4 demonstrates the benefits of parameter averaging in the setting of acoustic modeling, where it often leads to significantly reduced error rates. Intuitively, we can view the online, mistake-driven updates on individual utterances as stochastic gradient descent on the overall loss function. Parameter averaging smoothes out the randomness in this process. Fig. 6.2 illustrates this intuition by visualizing how

Table 6.4: Frame error rates from different forms of parameter averaging: no averaging, averaging in Φ by eq. (6.15), and averaging in Λ by eq. (6.16). See text for details.

# mix	Frame Error Rate (%)		
	no averaging	averaging in Φ	averaging in Λ
1	41.9	31.4	31.6
2	37.1	30.1	30.9
4	35.2	29.5	30.2
8	35.2	28.8	28.8
16	**33.5**	**28.6**	**28.4**
32	33.9	29.3	29.6

the GMM parameters across all states and mixture components evolve during training. In particular, for the purpose of visualization, the figure shows the two dimensional trajectory obtained by projecting these GMM parameters onto their first two principal components. Note how the averaged parameter estimates "spiral" down more quickly to the final solution.

As mentioned in section 6.5, for online, mistake-driven updates with the matrix factorization in eq. (6.13), there are two possible averaging procedures. The results show that better performance is generally (though not always) obtained by optimizing the matrices Λ_{sc} while averaging the matrices Φ_{sc}. We can offer one possible intuition for this trend. As noted earlier, the factorization in eq. (6.13) is not unique. Therefore we can imagine a sequence of parameter estimates that involve different values for Λ_{sc} but equal values for Φ_{sc}. (That is, the varying estimates for Λ_{sc} differ only by a unitary transformation.) In this case, the constant value of Φ_{sc} will be returned by averaging the matrices Φ_{sc} using eq. (6.12), but not by averaging the matrices Λ_{sc} using eq. (6.14). Though this is a contrived scenario unlikely to occur in practice, it suggests that averaging in Λ_{sc} can lead to nonsensical results.

Note that in eq. (6.15), we average from the very first parameter to the latest updated parameter. This might give too much weight to the first several parameters since they are not stable yet (see the points at the tail of the spiral in Fig. 6.2). An alternative to the averaging scheme is "moving average", which only looks back a fix number of points other than all history. The results are shown in Table 6.5. We performed the experiments

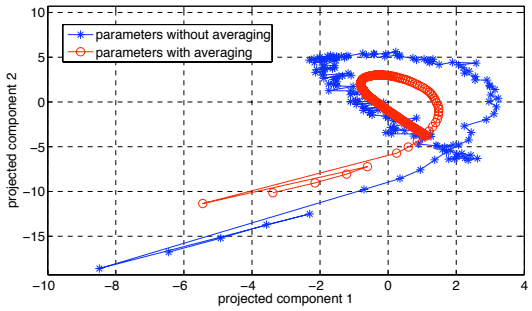

Figure 6.2: The trajectory of CD-HMM parameters during training. The figure visualizes the parameters Φ_{sc} by projecting them onto their first two principal components. Parameter averaging leads to faster convergence.

Table 6.5: Frame error rates from different lengths of parameter averaging: 10, 100, 1000, 2000. All are based on averaging in Φ by eq. (6.15) and updates in Λ by eq. (6.14). See text for details.

# mix	Frame Error Rate (%)					
	None	10	100	1000	2000	∞
1	41.9	39.3	36.6	33.9	32.4	31.4
2	37.1	36.6	35.3	33.0	32.2	30.1
4	35.2	35.2	34.7	32.0	31.2	29.5
8	35.2	34.5	32.6	31.4	31.1	28.8
16	**33.5**	**33.2**	**32.5**	**30.2**	**29.1**	**28.6**

with four different lengths of history - 10, 100, 1000, and 2000 past updates. Again the best model is selected by the validation set and marked by boldface.

Unlike averaging through all updates, moving average requires a temporary storage of all parameters in the limited length of history. When it comes to a complicated model, such as 32 and more mixture components for each state, the memory usage is considerable. We can see from Table 6.5 that the length of history does help the performance, and at some point the performance should meet the performance with infinite averaging. However, due to memory constraint, we cannot buffer more than 2000 parameters in the past updates.

Table 6.6: Frame error rates from differently initialized sets of model parameters, one set with zero values, the other with maximum likelihood (ML) estimates. The left results used the Φ-update in eq. (6.12); the right results used the Λ-update in eq. (6.14). See text for details.

#	Frame Error Rate (%)			
mix	$\Phi^{(0)}=0$	$\Phi^{(0)}=\Phi^{\mathrm{ML}}$	$\Lambda^{(0)}=0$	$\Lambda^{(0)}=\Lambda^{\mathrm{ML}}$
1	32.2	32.2	**33.1**	31.4
2	33.4	31.5	34.9	30.1
4	32.0	31.0	35.7	29.5
8	32.0	30.9	36.2	28.8
16	**31.6**	30.5	35.0	**28.6**
32	32.3	**30.4**	37.9	29.3

6.6.5 Benefits of initialization

For perceptron training in discrete HMMs, parameter values can simply be initialized as zeroes [Col02]. However, when GMMs are used in acoustic models, the discriminant function is not generally a linear function of the parameters, and the required optimization is not convex. Thus, depending on the quality of the initialization, the potential exists to get trapped in local minima.

Table 6.6 compares the results from online, mistake-driven updates using two different sets of initial model parameters: one set with zero values, the other with ML estimates. Two trends are clear. First, the ML initialization generally leads to better performance, especially as the model size is increased. Second, the zero-valued initialization leads to *worsening* performance with increasing model size when we update the parameters Λ_{sc} using eq. (6.14). These results suggest that the much faster convergence from the matrix factorization in eq. (6.13) comes at the expense of creating a more treacherous optimization.

6.7 Summary

In this chapter, we have explored various matrix updates for perceptron training of CD-HMMs. As our main contributions, we analyzed numerous issues of parameteri-

zation and smoothing that do not arise in perceptron training of discrete HMMs; we also performed systematic experiments in ASR to understand how these issues play out in practice. Our results show that not all forms of discriminative training are equally effective: indeed, matrix reparameterizations and factorizations can have a significant effect on classification performance as well as rates of convergence. In particular, the best performance of our approach was obtained by updates with the factorization in eq. (6.13), the reparameterizations in eq. (6.12), and the averaging in eq. (6.15).

Chapter 6, in part, is a reprint of the material as it appear in Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML-09) 2009. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, ACM, 2009. Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of these papers.

Chapter 7

Online Updates for Large Margin HMMs

In this chapter, we extend the online updates from the previous chapter to incorporate large margin constraints. The best known large-margin classifiers are support vector machines (SVM) [BGV92, CV95]. In binary classification problems, SVMs compute the maximal margin hyperplane that separates the positively and negatively labeled examples. These classifiers have strong theoretical guarantees on their generalization errors [Vap95].

Large-margin classifiers were originally formulated for classification of I.I.D data. In this decade, they have been explored for problems in sequence labeling [ATH03, TGK04]. In particular, a number of researchers in ASR have explored large margin methods for discriminative training of HMMs [JLL06a, LYL07, YDHA07, SS09]. Large margin training of HMMs seeks not only to minimize the empirical error rate, but also to separate the scores of correct and incorrect transcriptions by the largest possible amount, thus achieving better generalization on unseen data [TGK04, THJA04]. This idea has been independently investigated by many researchers in acoustic modeling and ASR [KSSB$^+$06, JLL06a, SS07a, LYL07, YDHA07]. Our main goal here is to investigate simple, online updates for large margin training of acoustic models.

7.1 Introduction

Recently, several researchers have proposed methods for large margin training of CD-HMMs [JLL06a, SS07a, LYL07, YDHA07, PKK+08, SP08]. In large margin training, acoustic models are estimated to assign significantly higher scores to correct transcriptions than competing ones; in particular, the margin between these scores may be required to grow in proportion to the total number of recognition errors [SS07a, PKK+08, SP08]. Empirically, large margin training has improved the performance of many systems beyond other leading discriminative approaches.

Large margin training in CD-HMMs has the same basic computational requirements as other discriminative approaches. The updates depend on computing statistics of hidden states as well as gradients with respect to various model parameters. For each update, these quantities must be computed and accumulated over all the utterances in the training corpus. To cope with large corpora, researchers often parallelize this batch computation across many different nodes, then combine the individual results as needed to average over all the training utterances [VOWY97].

In this chapter, we investigate a different, simpler approach for accelerating large margin training of CD-HMMs. We replace the batch computation described above by an *online, sequential* computation based on the mistake-driven algorithm in Chapter 6. Specifically, we optimize the CD-HMM parameters in an incremental fashion, updating them after the decoding of each training utterance. We find that this approach converges much more quickly than previously developed batch optimizations of large margin CD-HMMs [SS07a]. We also find that it yields significantly more accurate acoustic models than other approaches—both online and batch—that do not attempt to enforce a large margin [CSS09c].

7.2 Large margin training

Let $(\mathbf{x}, \mathbf{y})$ denote an observation sequence and its ground truth transcription. The essence of large margin training lies in the following observation: whereas for correct recognition we merely require the inequalities in eq. (6.2), for correct recognition *by a*

large margin, we additionally require that

$$\forall \mathbf{s} \neq \mathbf{y}, \quad \mathcal{D}(\mathbf{x}, \mathbf{y}) > \mathcal{D}(\mathbf{x}, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}), \tag{7.1}$$

where $\mathcal{H}(\mathbf{s}, \mathbf{y})$ is the Hamming distance between two hidden state sequences of the same length, and $\rho > 0$ is a constant margin scaling factor. In other words, for large margin training, the score of the correct transcription should exceed the score of any incorrect transcription by an amount that grows in proportion to the number of recognition errors.

We can use dynamic programming to compute the hidden state sequence that most egregiously violates the margin constraint in eq. (7.1). We use $\tilde{\mathbf{s}}^*$ to denote this hidden state sequence. From eq. (7.1), we have:

$$\tilde{\mathbf{s}}^* = \text{argmax}_{\mathbf{s} \neq \mathbf{y}} \left[\mathcal{D}(\mathbf{x}, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}) \right]. \tag{7.2}$$

The right hand side of eq. (7.2) can be maximized by a simple variant of the standard Viterbi algorithm [**?**]. We emphasize that the margin-based decoding selects and penalizes incorrect sequences that are *close* in log-likelihood but *far away* in Hamming distance. Put another way, if two competing sequences have the same log-likelihood, then the margin-based decoding will select and penalize the one with more (frame-level) transcription errors.

To measure the total amount of constraint violation in eq. (7.1), we define the loss function:

$$\mathcal{L}(\Theta) \; = \; \sum_n \left[\max_{\mathbf{s} \neq \mathbf{y}} \left[\mathcal{D}(\mathbf{x}, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}) \right] - \mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) \right]^+, \tag{7.3}$$

analogous to the loss function in eq. (6.3). For online training of large margin HMMs, we consider the following update rule:

$$\Theta \leftarrow \Theta + \eta \frac{\partial}{\partial \Theta} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \tilde{\mathbf{s}}_n^*) \right]. \tag{7.4}$$

The update is applied whenever the margin-based decoding in eq. (7.2) yields a state sequence that violates the inequality in eq. (7.1). Eq. (7.4) differs from eq. (6.4) in one critical aspect: namely, we replace the usual Viterbi sequence in eq. (5.3) by the sequence from margin-based decoding in eq. (7.2). This substitution changes the nature of the optimization in an important way: even if an utterance is correctly decoded,

eq. (7.4) may still update the model parameters. In particular, the parameters will be updated if there exists an incorrect decoding whose log-likelihood is not sufficiently well separated from that of the correct transcription.

Though the margin scaling factor ρ does not appear explicitly in eq. (7.4), it directly affects the computation of $\tilde{\mathbf{s}}_n^*$. In fact, our experiments will show that the subtle change in eq. (7.4) leads to profoundly different updates.

The online update in eq. (7.4) is written in terms of the parameters Θ of the CD-HMM. In this model, we adopt the same parameterization of CD-HMMs that has proven useful in the earlier models (sections 6.4 and 6.5). Also, in all the following experiments, we only adapt the parameters of the GMMs, not the transition probabilities of the CD-HMMs. To obtain smoother parameter estimates over time, the results from eq. (7.4) can also be averaged as in eq. (6.5). We performed this averaging in all of our experiments.

7.3 Experiments

Following the same experimental set-up as in previous chapters, we sought to investigate the potential benefits of online updates for large margin training. All CD-HMMs were initialized by ML estimation. Starting from these baseline CD-HMMs, we then compared the performance of the different online updates in eq. (6.4) and (7.4). For these comparisons, we used the parameterization in eq. (6.13) and the averaging in eq. (6.5), since these choices yielded the best results for online updates without large margin constraints. For the margin-based update, the results of training depend on the margin scaling factor ρ. We experimented with a wide range of values for this scaling factor.

7.3.1 Benefits of online large margin training

Table 7.1 shows the results from the best models trained in this way. (For the margin-based results, we chose the scaling factor ρ that yielded the lowest phone error rates on the held-out development set.) The results show that online updating with margin-based decoding significantly reduces the frame and phone error rates across all

model sizes. In general, the frame error rates improve more than the phone error rates; this discrepancy reflects the fact that the margin-based updates more closely track the Hamming distance (not the edit distance) between target and Viterbi phone sequences. Nevertheless, comparing to the results in Table 6.1, we see that the gains from margin-based decoding exceed the gains from all other methods (batch and online) that do not incorporate large margin constraints.

For reference, table 7.1 also reproduces previous results obtained from batch implementations of large margin training [SS09]. The objective function for batch training differed slightly from the ones we use in eq. (7.3) for online learning; specifically, the batch optimization minimized a soft-max approximation to the first term in eq. (7.3) using a projected subgradient method. The online updates do not quite match the performance of the batch implementation; however, they are simpler to implement and require fewer passes through the set of training utterances. Moreover, we will see in section 8.1 that the online updates can be extended in simple ways to achieve even further gains.

While Table 7.1 quantifies the effects of margin-based decoding on error rates, Fig. 7.1 graphically illustrates the profound influence it exerts during training. To create this figure, we computed the Hamming distance between the Viterbi decoding $\mathbf{s}^*$ in eq. (5.3) and the margin-based decoding $\tilde{\mathbf{s}}^*$ in eq. (7.2) for each utterance during one online pass through the training corpus. The figure shows a histogram of these Hamming distances after they have been normalized by the number of frames in the utterance. The histogram's peak away from zero shows that margin-based decoding yields very different competing transcriptions for discriminative training than standard Viterbi decoding.

The frame and phone error rates from large margin training depend on the value of the margin scaling factor ρ. Fig. 7.2 shows this dependence for HMMs with 4-component GMMs in each state. More generally, for phone error rates on the development set, the optimal values of ρ were respectively 0.8, 1.0, 0.7, and 1.0 for HMMs with 1, 2, 4, and 8-component GMMs. Training with $\rho = 0$ (i.e., without margin-based decoding) produces the results shown in the middle columns of Table 7.1.

Fig. 7.3 illustrates the relatively fast convergence of online learning. The figure shows the frame error rates on the development data set during training. For all model sizes, most of the improvement from online learning occurs during the first 10-20 passes

Table 7.1: Frame error rates (*top*) and phone error rates (*bottom*) on the TIMIT test set for acoustic models of varying size, as obtained by maximum likelihood (ML) estimation, online updates with standard Viterbi decoding, online updates with margin-based decoding, and a batch implementation of large margin training [SS09].

# mixture component	Frame Error Rate (%)			
	Maximum likelihood	Online w/o margin	Online w/ margin	Batch w/ margin
1	39.7	31.4	30.5	29.5
2	36.2	30.1	29.4	29.0
4	33.1	29.5	28.3	28.4
8	30.7	28.8	27.3	27.2
16	**29.5**	**28.6**	**27.3**	
32	29.9	29.3	27.6	

# mixture component	Phone Error Rate (%)			
	Maximum likelihood	Online w/o margin	Online w/ margin	Batch w/ margin
1	41.5	33.6	32.8	31.2
2	38.0	32.3	31.4	30.8
4	34.9	31.4	30.3	29.8
8	32.3	30.1	28.6	28.2
16	**30.8**	**29.7**	**28.8**	
32	31.8	30.9	29.0	

through the training corpus. Many more passes are typically required for convergence of batch methods.

7.3.2 Hamming distance reweighting

The Hamming distance in eq. (7.1) is simply the frame-wise difference between two sequences. Thus it implicitly treats all phoneme substitution errors as equally significant. However, that is not true. For example, the difference between the phonemes /m/, /a/ (consonant and vowel) is perceptually stronger than that between /m/, /n/ (consonant and consonant). This observation inspired us to explore alternative measures of distance between phoneme sequences.

One idea that we explored is to compute a *weighted* Hamming distance between

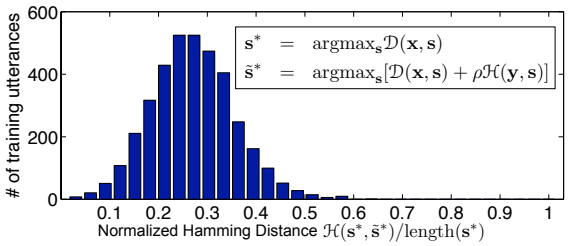

Figure 7.1: Histogram of normalized Hamming distances between sequences from Viterbi and margin-based decoding. The distances were computed during the fifth iteration through the training corpus for the best-performing large margin HMM with sixteen Gaussian mixture components per hidden state.

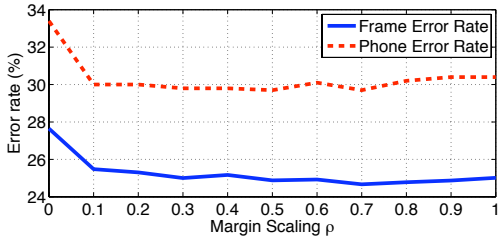

Figure 7.2: Frame and phone error rates on the development set as a function of the margin scaling factor ρ. Results are shown for acoustic models with four Gaussian mixture components per hidden state.

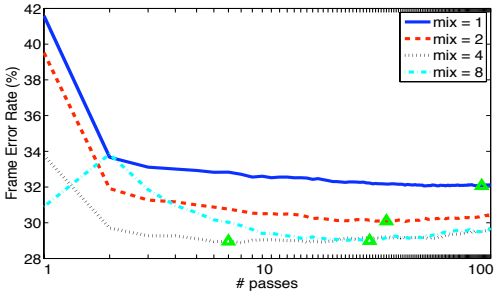

Figure 7.3: Frame error rates on the development set during training. The triangles mark the best models obtained for different numbers of Gaussian mixture components.

phoneme sequences. Intuitively, the weightings should depend on some sorts of similarity measure between phonemes. This gives rise to two major questions: 1) How do

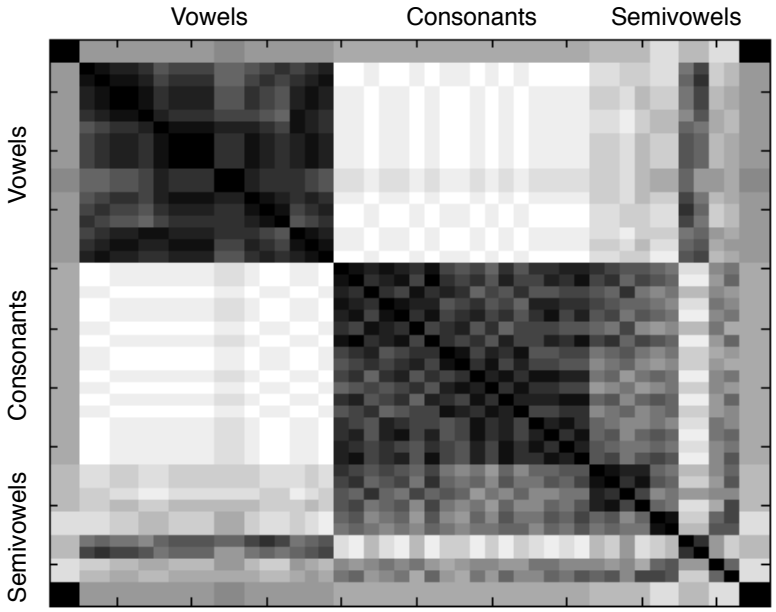

Figure 7.4: The distance matrix of phonemes in TIMIT dataset, based on distinctive feature compositions. The matrix is represented in a gray scale image, in which black color means distance of 0.

we weight the phoneme substitution? Should we penalize or promote substitutions between closer phonemes? 2) Since phonemes are abstract objects, how do we define the distance between them?

As discussed in Section 7.2, in large margin sequence classifiers we want to compete with the strongest violator, and push the decision boundary away from it. Therefore, in eq. (7.1) we favor a sequence with a larger Hamming distance as the competitor. Similarly here, we weight the phoneme substitution by the "distance" between phonemes in order to favor a sequence which is very different from the correct transcription *perceptionally*. The computation of weighted Hamming distance involves a distance matrix of phonemes and a weighted count of the difference between two sequences.

Here we explore two metrics based on phonological rules from studies of human speech, production, and perception. The first metric is based on *distinctive features* [CH68]. These are 22 unique phonological rules which are sufficient to analyse

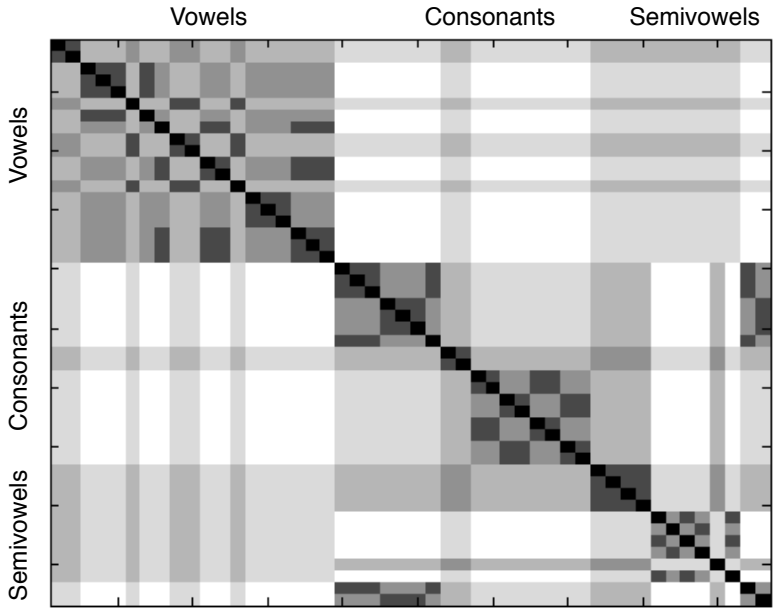

Figure 7.5: The distance matrix of phonemes in TIMIT dataset, based on phonological tree. The matrix is represented in a gray scale image, in which black color means distance of 0.

base units (e.g, phonemes) of any languages. In the distinctive feature representation, each phoneme is composed of binary phonological features, denoted by + and - signs. The encoding can be displayed as a matrix, where each column represents a phoneme and each row represents a feature. The matrices for vowels and consonants in English are shown in Table 7.2 and 7.3; for a complete table, see [CH68, Bn98]. Note that some phonological features are left blank in the matrix, which represent "either + or -". Since the distinctive features are universal phonological features (capable of encoding any languages), some of them are not observed in English. In practice, the phonological feature set may be changed by the regional effects in each language and the composition of any particular dataset. We adopt a modified feature composition for TIMIT dataset reported in [Bn98, LW01].

Table 7.2: Distinctive feature matrix for *vowels* in English. The phonemes are denoted in the labels of both TIMIT dataset and International Phonetic Alphabet (IPA). Each row represents one distinctive feature, and the value are indicated by + and - signs. Blank values mean "either + or -". For the full phonetic definition of + and - signs and the full list of sounds, refer to [CH68, Bn98].

TIMIT	IY	IH	EH	EY	AE	AA	AW	AY	AH
sonorant	+	+	+	+	+	+	+	+	+
syllabic	+	+	+	+	+	+	+	+	+
consonantal	-	-	-	-	-	-	-	-	-
high	+	+	-	-	-	-	-	-	-
back	-	-	-	-	-	+	+	+	+
front	+	+	+	+	+	-	-	-	-
low	-	-	-	-	+	+	-	-	-
round	-	-	-	-	-	-	-	-	-
tense	+	-	-	-	-	-	-	-	-
anterior									
coronal									
voice									
continuant									
nasal									
strident									

TIMIT	AO	OY	OW	UH	UW	ER	AX	IX
sonorant	+	+	+	+	+	+	+	+
syllabic	+	+	+	+	+	+	+	+
consonantal	-	-	-	-	-	-	-	-
high	-	-	-	+	+	-	-	+
back	+	+	+	+	+	-	-	-
front	-	-	-	-	-	-	-	-
low	+	+	-	-	-	-	-	-
round	+	+	+	+	+	+	-	-
tense			-	-	+		-	-
anterior								
coronal								
voice								
continuant								
nasal								
strident								

Table 7.3: Distinctive feature matrix for *consonants* in English [CH68, Bn98].

TIMIT	B (BCL)	D (DCL) (DX)	G (GCL)	P (PCL)	T (TCL) (Q)	K (KCL)	JH	CH	S	SH	Z	ZH	F
IPA	/b/	/d/	/g/	/p/	/t/	/k/	/dZ/	/tS/	/s/	/S/	/z/	/Z/	/f/
sonorant	-	-	-	-	-	-	-	-	-	-	-	-	-
syllabic	-	-	-	-	-	-	-	-	-	-	-	-	-
consonantal	+	+	+	+	+	+	+	+	+	+	+	+	+
high	-	-	+	-	-	+	+	+	-	+	-	+	-
back													
front													
low													
round													
tense													
anterior	+	+	-	+	+	-	-	-	+	-	+	-	+
coronal	-	+	-	-	+	-	+	+	+	+	+	+	-
voice	+	+	+	-	-	-	+	-	-	-	+	+	-
continuant	-	-	-	-	-	-	-	-	+	+	+	+	+
nasal	-	-	-	-	-	-	-	-	-	-	-	-	-
strident	-	-	-	-	-	-	+	+	+	+	+	+	+

TIMIT	TH	V	DH	W	Y	HH (HV)	R	L	M	N (NX)	NG	EL
IPA	/T/	/v/	/D/	/w/	/y/	/h/	/r/	/l/	/m/	/n/	/N/	/L/
sonorant	-	-	-	+	+	+	+	+	+	+	+	+
syllabic	-	-	-	-	-	-	-	-	-	-	-	+
consonantal	+	+	+	-	-	-	+	+	+	+	+	+
high	-	-	-	+	+	-	-	-	-	-	+	-
back				+	-	-	-	-				-
front				-	+	-	-	-				-
low				-	-	+	-	-	-	-	-	-
round				+	-							
tense				-	-							
anterior	+	+	+	-	-	-	-	+	+	+	-	+
coronal	+	-	+	-	-	-	+	+	-	+	-	+
voice	-	+	+			-	+	+	+	+	+	+
continuant	+	+	+			+	+	+	-	-	-	+
nasal	-	-	-			-	-	-	+	+	+	-
strident	-	+	-			-	-	-	-	-	-	-

Table 7.4: Frame error rates (*top*) and phone error rates (*bottom*) on the TIMIT test set for acoustic models of varying size, as obtained by maximum likelihood (ML) estimation, online updates with margin-based decoding, and online updates with reweighted margin-based decoding based on two types of distance measures - distinctive features (DF), and phonological tree (PT).

# mixture component	Frame Error Rate (%)			
	Maximum likelihood	Online w/ margin	Reweighting DF	Reweighting PT
1	39.7	30.5	30.5	30.7
2	36.2	29.4	29.0	29.6
4	33.1	28.3	28.5	28.1
8	30.7	27.3	27.7	27.7
16	**29.5**	**27.3**	**27.0**	**27.4**
32	29.9	27.6	27.7	27.1

# mixture component	Phone Error Rate (%)			
	Maximum likelihood	Online w/ margin	Reweighting DF	Reweighting PT
1	41.5	32.8	32.5	32.9
2	38.0	31.4	30.8	31.2
4	34.9	30.3	30.0	30.3
8	32.3	28.6	29.1	29.4
16	**30.8**	**28.8**	**28.5**	**29.3**
32	31.8	29.0	29.3	28.6

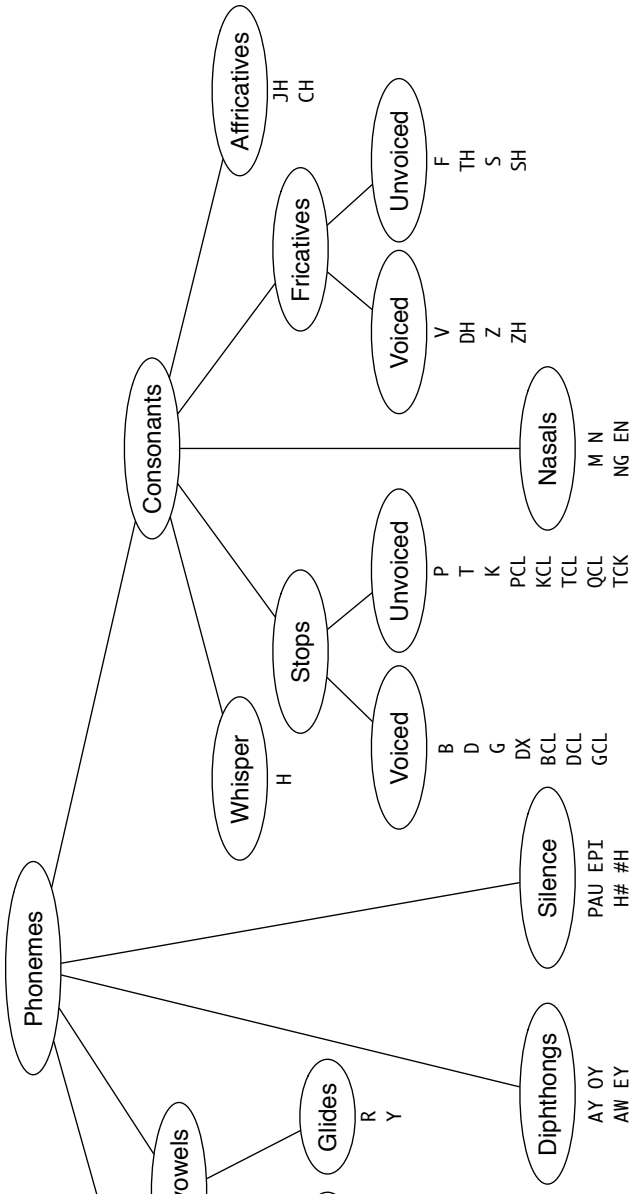

Figure 7.6: The hierarchical tree structure of 65 phonemes in TIMIT dataset. Each node represent a phonological feature, and upper nodes are major class features. The categorization is based on [RJ93].

The distance between two phonemes is defined as the difference between the distinctive features of the two phonemes. The resulting distance matrix of phonemes in TIMIT dataset is shown in Fig. 7.4. The distance matrix is illustrated as a gray-scale image, in which black denotes distance of 0. The distance in general follows our intuitions: phonemes in the same broad category (e.g. consonants) have smaller distances. We perform the experiments by weighting the substitutions in Hamming distance with the distance matrix, and train the model parameters by optimizing eq. (7.3). Again we use validation set to select the best scaling factor λ. The results are shown in Table 7.4. The Hamming distance reweighting yields marginal improvements on both frame and phone error rates.

The second distance measure we explore for reweighting Hamming distances is based on the phonological tree [RJ93]. The corresponding tree structure for TIMIT dataset is shown in Fig. 7.6, in which each intermediate node can be viewed as a phonological feature and each leaf is a phoneme. The leaf nodes under the same ancestor share more phonological propertiesthan others. Therefore, we define the distance as the total distance from the two leaves to their lowest common ancestor (LCA). For example, the phonemes inside the same category "front vowels" have a distance of 2, and those between "front vowels" and "mid vowels" have a distance of 4. Again from this reweighted Hamming distance, we can represent the distance matrix by a gray-scale image (see Fig. 7.5). The results are shown in the last column of Table 7.4. However, the reweighting based on the phonological tree does not improve the performance.

7.3.3 Forced alignment

Finally, we consider the applicability of our approach to other common training scenarios. While the TIMIT speech corpus has manually aligned phonetic transcriptions, most speech corpora do not have such information. For large margin training, our framework requires target state sequences that specify the hidden state in each frame of speech. When these alignments are not available from the corpus itself, what can we use in their place? The simplest option is to compute forced alignments of the training speech from whatever word or phonetic transcriptions are provided. To evaluate this option, we experimented with large margin training where for target state sequences,

we used forced alignments generated by seed CD-HMMs trained by ML estimation. Table 7.5 compares the phone error rates from large margin training using manual versus forced alignments for different model sizes. Surprisingly, the results from forced alignments are comparable to (and sometimes even slightly better than) those obtained from the "ground truth" transcriptions. Thus it does not seem that precise knowledge of phoneme boundaries is required for large margin training, provided that forced alignments of reasonable quality can be generated from a seed model.

Table 7.5: Phone error rates from large margin training using manually aligned phonetic transcriptions versus forced alignments; see text for details.

#	Phone Error Rate (%)	
mix	manual	forced
1	32.8	32.9
2	31.4	30.9
4	30.3	29.8
8	28.6	28.9
16	**28.8**	**28.2**
32	29.0	30.6

7.3.4 Diagonal covariance matrices

Our final experiment was motivated by the fact that many researchers choose not to use full covariance matrices in CD-HMMs for ASR. Table 7.6 compares the results when HMMs with diagonal covariance matrices were estimated by ML versus online updates for large margin training. For the latter, we used slight variants of the online updates in eqs. (6.6)-(6.8); in particular, we constrained the covariance matrices to be diagonal, and we computed gradients with respect to the large-margin loss function in eq. (7.3). The results show that for purely diagonal covariance matrices, large margin training also yields lower frame and phone error rates than ML estimation. However, the improvements are not as substantial as those obtained from full covariance matrices using the parameterization in eq. (6.10).

Table 7.6: Frame and phone error rates for HMMs with diagonal covariance matrices, as obtained by maximum likelihood (ML) estimation and online updates for large margin training. The left column shows the number of mixture components per GMM.

#	Frame Error Rate (%)		Phone Error Rate (%)	
mix	ML	online	ML	online
1	44.0	39.2	46.8	43.5
2	40.0	35.6	43.3	39.3
4	37.6	34.0	41.1	37.1
8	34.8	32.3	37.5	35.3
16	34.1	31.0	36.4	34.1
32	32.5	31.0	34.5	33.0
64	31.5	31.4	33.5	33.5
128	30.6	**30.2**	32.8	**32.0**
256	31.6	31.4	33.6	33.4

7.4 Summary

Online learning is an active area of research in machine learning [BL04, BB08]. Our main contribution in this chapter lies in adapting various recent approaches [Col02, CSS09c] to large margin training of CD-HMMs. On TIMIT phoneme recognition, we have shown that our approach is effective and efficient, not only attaining better error rates than standard batch algorithms [SS09], but also speeding up training time significantly. Anecdotally, we have attained similar performance as our own batch implementation of large margin training in roughly one third of the training time.

Scaling our approach to large vocabulary ASR presents several challenges. Online algorithms tend to update parameters very aggressively, thus exploring the parameter space more quickly than batch algorithms but also exhibiting larger variance on consecutive updates. Future work will explore how to balance these tendencies. One possible strategy is to chunk large amounts of data into small subsets, then to update the model parameters using statistics on subsets as opposed to individual utterances. This "minibatch" scheme lends itself naturally to parallelization since the computations on subsets of utterances can be distributed across multiple machines. Within this approach, however, further research is needed to determine the optimal subset size.

Chapter 7, in part, is a reprint of the material as it appear in Proceedings of

the Tenth Annual Conference of the International Speech Communication Association (Interspeech-09) 2009. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, International Speech Communication Association (ISCA), 2009. Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of these papers.

Chapter 8

Acoustic Feature Adaptation

Modern systems for automatic speech recognition (ASR) consist of two interrelated components: a *front end* for signal processing and feature extraction, and a *back end* for statistical inference and pattern recognition. In most systems, the front end computes mel-frequency cepstral coefficients (MFCCs) and higher-order derivatives of MFCCs that capture changes over time [RJ93]. The back end then analyzes and interprets these MFCCs using continuous-density hidden Markov models (CD-HMMs). While the parameters of CD-HMMs are estimated from large amounts of speech, it is less common for the parameters in the front end to be systematically optimized in the same way.

In this chapter we continue a general line of research for end-to-end training of speech recognizers. Specifically, we focus on the large-margin training framework discussed in Chapter 6 and 7 of CD-HMMs and apply large-margin methods to jointly optimize the acoustic features computed by the front end along with the CD-HMM parameters estimated by the back end.

8.1 Introduction

Many researchers have noted the discrepancy between front and back ends and shown that feature space adaptation indeed improves the performance by a significant amount. In particular, adaptive methods are increasingly being applied at all stages of pattern recognition, from the lowest levels of feature extraction to the highest levels of

decision-making. Early influential work along these lines involved data-driven methods for robust feature extraction [SEK+00] and filterbank design [MH03, SWH+89, HW91]. More recent methods include: (i) heteroscedastic linear discriminant analysis (HLDA) [KA98] and neighborhood component analysis [SMCH07] to learn informative low dimensional projections of high dimensional acoustic feature vectors; (ii) stochastic gradient and second-order methods to tune parameters related to frequency warping and mel-scale filterbanks [VG04, BVG04]; (iii) maximum likelihood methods for speaker and environment adaptation [Gal98a, SFK+05] that perform linear transformations of the acoustic feature space at test time; and (iv) extensions of popular frameworks for discriminative training, such as minimum phone error [ZM05] and maximum mutual information [MWS08], to learn accuracy-improving transformations and projections of the acoustic feature space.

In this chapter, we show how to extend the large margin updates in eq. (7.4) to learn a linear transformation of the acoustic feature space. The linear transformation is parameterized by a projection matrix which maps the cepstral coefficients from multiple adjacent analysis windows into a lower-dimensional acoustic feature vector. We derive online updates to adapt the elements of this projection matrix after the decoding of each training utterance. Our approach can also be viewed as a strategy for learning low-rank decomposition of very large covariance matrices [WBS06, SMCH07].

The projection matrix affects how acoustic feature vectors are computed in every frame of speech. For this reason, small changes to the projection matrix can have large effects on recognition. This sensitivity presents a challenge for online learning, where the acoustic feature space is constantly adapted based on the statistics of individual training utterances. To mitigate the strongly biased gradients from individual utterances, we experimented with different schemes for regularization and parameter-tying. Our results show that parameter-tying helps to stabilize online learning by accumulating and averaging gradients across otherwise independent computations.

Our work is distinguished from previous schemes for feature adaptation in three ways. First, we consider how to jointly optimize the parameters in the front end along with the acoustic models in the back end. Second, the feature adaptation is driven by an objective function for large margin training, which seeks to separate the log-likelihoods

of correct and incorrect transcriptions by an amount proportional to their Hamming distance. Third, we explore parameter-tying not across different mixture components or hidden states in the same HMM, but across different recognizers; in particular, we train several different recognizers in parallel while tying the feature projection matrices in their front ends.

8.2 Derivative features and linear projections

In most systems for ASR, the front end computes acoustic feature vectors from mel-frequency cepstral coefficients (MFCCs). Typically, the first $d_0 = 13$ MFCCs are used in this analysis. Due to co-articulation and other temporal effects, the MFCCs from one analysis window may contain information about the phonetic content in neighboring windows. To capture this information, most front ends also incorporate MFCCs from neighboring windows into their acoustic feature vectors. In particular, they compute derivative features, such as delta and delta-delta MFCCs, and augment the feature vector to include them.

The derivative features are computed by linearly combining MFCCs from neighboring analysis windows. The weights used to combine adjacent MFCCs are fixed and determined heuristically. Unlike most other parameters in modern speech recognizers, these weights in the front end are not typically adapted to optimize performance.

To start, we consider how to optimize the linear transformation used to compute derivative features in conjunction with the back end for large margin HMMs (described in section 7.2). The standard derivative features are computed from a linear transformation of the raw MFCCs in nearby frames. Let u_t denote the $d_0 = 13$ MFCCs computed at time t, and let v_t denote the "stacked" MFCCs obtained by concatenating $4K + 1$ consecutive frames $u_{t-2K}, u_{t-2K+1}, ..., u_t, ..., u_{t+2K}$ for some small value of K. The first-order and second-order derivative features δ and Δ are computed by

$$\delta_t = \frac{\sum_{n=-K}^{K} n \cdot u_{t+n}}{\sum_{n=-K}^{K} n^2} \tag{8.1}$$

and

$$\Delta_t = \frac{\sum_{n=-K}^{K} n \cdot \delta_{t+n}}{\sum_{n=-K}^{K} n^2}. \tag{8.2}$$

Note that for computing these derivative features, there are many possible choices for the number of neighboring windows and linear combination weights. It is straightforward to compute higher order derivative features; however, such higher-order features provide only marginal improvements in performance for most ASR.

Finally, let x_t denote the acoustic feature vector derived from the MFCCs at time t and their first and second-order derivatives. Then x_t and v_t can be related by the linear transformation:

$$x_t = H_0 v_t, \tag{8.3}$$

where H_0 is the projection matrix whose entries approximate derivatives by finite differencing operations on nearby frames. Note that eq. (8.3) describes how acoustic feature vectors were computed for all the experiments described in previous sections of this thesis.

The matrix H_0 is only one of many possible projection matrices that can be used to compute acoustic feature vectors from MFCCs in adjacent frames of speech. By learning projections other than the finite differencing operations in eqs. (8.1-8.2), we hope to improve on previous results and benchmarks. In fact, both H_0 and v_t can be extended to more general settings. For example, v_t is usually defined as a context window of 13 or more consecutive frames in modern speech systems [SRP09], or v_t can be spliced log-spectral features. In the case of raw spectral features, H_0 can be initialized as the discrete cosine transform (DCT) for mel-frequency feature computation as described in Chapter 4.

Our ultimate goal here is to learn more general projection matrices in the context of large margin training for HMMs. In particular, we are interested in exploring a different and adaptive linear combination on top of the traditional MFCC features.

8.3 Loss function for feature adaptation

Our approach builds on the online updates for large margin training in eq. (7.4). Let $\hat{x}_t$ denote the augmented feature vector of the stacked feature vector v_t of MFCCs

from $4K + 1$ adjacent windows, as described in section 8.2:

$$\hat{x}_t = \begin{bmatrix} v_t \\ 1 \end{bmatrix}, \tag{8.4}$$

and let z_t denote the lower dimensional acoustic feature vector that appears in eq. (6.11). We seek a projection matrix $H \in \Re^{D \times d}$ that maps the high-dimensional vector $\hat{x}$ of stacked MFCCs to the low-dimensional acoustic feature vector z_t; then for each window, we can compute:

$$z_t = H\hat{x}_t. \tag{8.5}$$

Note that H has one extra row and column than the projection matrix H_0 in eq. (8.3) due to the augmented feature vector z that appears in eq. (6.11) for large margin HMMs. In particular, we have $d = 3d_0 + 1$ and $D = (4K+1)d_0 + 1$, where $d_0 = 13$ is the number of MFCCs computed per window.

For acoustic feature adaptation in large margin HMMs, we update the projection matrix H and the parameter matrices Φ_{sc} so that the constraints in eq. (7.1) are satisfied for as many training utterances as possible. Let $\{(\hat{\mathbf{x}}_n, \mathbf{y}_n)\}_{n=1}^N$ denote the N labeled feature-state sequences in the training corpus, where the observations live in the high-dimensional feature space (before projection). For online learning, we examine one utterance at a time and compute the hidden state sequence by eq. (7.2). Analogous to sections 6.2 and 7.2, we define the loss function as:

$$\mathcal{L}(H, \Phi) = \sum_n \left[\max_{\mathbf{s} \neq \mathbf{y}_n} \left[\mathcal{D}(\hat{\mathbf{x}}_n, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}_n) \right] - \mathcal{D}(\hat{\mathbf{x}}_n, \mathbf{y}_n) \right]^+. \tag{8.6}$$

Eq. (8.6) differs from eq. (7.3) in two respects: first, the observed feature vectors in this context live in a much higher-dimensional space; second, in addition to the model parameters Φ, the loss function also depends on the feature projection matrix H.

The margin-based loss function in eq. (8.6) depends on the matrices Φ_{sc} and H through eqs. (6.1, 6.11) and (8.5). Specifically, we can write the discriminant function as:

$$\begin{aligned} \mathcal{D}(\hat{\mathbf{x}}, \mathbf{s}) \;\; &= \;\; \log \mathcal{P}(s_1) \;+\; \sum_{t=1}^{T-1} \log \mathcal{P}(s_{t+1}|s_t) \\ &\quad + \sum_{t=1}^{T} \log \sum_c e^{-\frac{1}{2}\hat{x}_t^\top H^\top \Phi_{sc} H \hat{x}_t}. \end{aligned} \tag{8.7}$$

Note that while eq. (8.7) depends on the high dimensional (stacked) cepstral feature vectors $\hat{x}_t \in \Re^D$, the loss function can be computed entirely in terms of the low dimensional features $z_t = H\hat{x}_t$. In fact, we can view $H^\top \Phi_{sc} H$ as storing a low-rank factorization of an inverse covariance matrix in the high dimensional space of unprojected cepstral features.

8.4 Low-rank factorization

The exponent in the last term in eq. (8.7) is essentially defining a distance function from example $\hat{x}_t$ to class centroid Φ_{sc} on the projected space by

$$\text{Dist}_{sc}(\hat{x}_t) = \hat{x}_t^\top H^\top \Phi_{sc} H \hat{x}_t = \hat{x}_t^\top H^\top \Lambda_{sc}^T \Lambda_{sc} H \hat{x}_t = \hat{x}_t^\top Q_{sc}^\top Q_{sc} \hat{x}_t, \tag{8.8}$$

where $Q_{sc} = \Lambda_{sc} H$, and by setting $d \leq D$, Q_{sc} is a low-rank factorization of the distance metric $Q_{sc}^T Q_{sc}$ of cluster c of state s.

Low-rank factorization is the key to many linear dimensionality reduction techniques such as linear discriminant analysis (LDA). It is often beneficial to reduce dimensionality for classification in the sense that the computation at inference phase is faster and also avoids overfitting at the training phase.

One might think by decomposing the projection matrix $H^\top \Phi_{sc} H$ into $Q_{sc}^\top Q_{sc}$, we can minimize the loss function in eq. (8.7) by an update similar to eq. (6.14), except the matrices are not square any more. However, this approach is ill-advised for several reasons.

First, when computing the gradient in eq. (6.14), we need to calculate the outer product of each example $\hat{x}_t \hat{x}_t^\top$ (see Appendix A for details), and since $\hat{x}_t$ inhabits a very high dimensional space, the computation is costly in both memory and CPU time.

Second, the low-rank matrix Q_{sc} is not a global projection matrix as opposed to the projection matrix H in eq. (8.7) and many of its counterparts (e.g, LDA, neighbourhood component analysis (NCA) [GRHS05], constrained maximum likelihood linear transformation [Gal98a]). A global projection matrix for feature vectors is often useful. For example in the case of speaker adaptation, the model can be adapted to each speaker by learning a transformation matrix for the speaker, without changing the original model distributions.

Instead, we derive updates separately for Λ_{sc} and H.

8.5 Parameter-tying

The loss function in eq. (8.6) can be minimized by alternately updating H and Φ_{sc}. However, we have noticed that small changes in the projection matrix H can drastically change the decoding results. This sensitivity is to be expected since the projection matrix H is used to calculate acoustic features in every frame of speech.

One way to reduce this sensitivity is to perform some sort of averaging. Batch training reduces this sensitivity by averaging over all the utterances in the training set. However, batch training does not exploit the fact that many training utterances convey redundant information. Some of the advantages of batch training can be obtained by online updates that average gradients over "mini-batches" of training utterances. For acoustic feature adaptation, however, we found that additional measures were needed.

The rest of this section describes a parameter-tying scheme that helps to mitigate the strongly biased gradients from individual training utterances. In this scheme, we tie the projection matrix H across several different recognizers whose parameters are jointly updated after decoding each training utterance. By averaging the gradients across multiple recognizers, we hope to obtain more stable online updates.

Parameter-tying in CD-HMMs has been widely adopted for
ASR [You92, DM94]. It has two main benefits: first, it reduces the memory footprint of speech recognizers, and second, it reduces the number of free parameters that must be estimated from limited training data. Our scheme for parameter-tying is subtly different than previous approaches. Typically, parameters are tied across different hidden states or mixture components in the same recognizer. For example, the related work [You92] used a data-driven clustering procedure to tie similar states among triphone models in the recognizer, where the similarity is determined by the Euclidean distance between the state means. In our scheme, however, we tie parameters across multiple different recognizers that are trained in parallel. These recognizers may have different model sizes (i.e., different numbers of hidden states and/or mixture components); or more generally, we consider all recognizers sharing the same feature space, such as the HMMs in context

dependent models. By tying the projection matrix, however, we force all the recognizers to use the same front end.

Our approach is based on a global cost function for parallel training of multiple models or recognizers. We index each available model by the superscript α; thus, each model has its own (back-end) parameters Φ^α, as well as a shared (front-end) feature projection matrix H. The global cost function is given by:

$$\mathcal{L} \;=\; \sum_{\alpha,n} \left[\max_{\mathbf{s} \neq \mathbf{y}_n} \left[\mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}_n) \right] - \mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{y}_n) \right]^+ , \tag{8.9}$$

where $\mathcal{D}^\alpha(\hat{\mathbf{x}}, \mathbf{s})$ is the discriminant function for the model with parameters Φ^α. In our implementation, the available models are large margin HMMs with one hidden state per phone but different numbers of Gaussian mixture components per hidden state. Eq. (8.9) differs from eq. (8.6) only in the accumulation of information across models. Thus, the parameter-tying only affects the gradients for optimizing the tied projection matrix H, but not the gradients for optimizing each model's individual (non-tied) parameter matrix Φ^α.

8.6 Online updates

The objective function in eq. (8.9) lends itself to an alternating minimization procedure. Such a procedure alternates between two phases, one optimizing Φ while holding H fixed; the other optimizing H while holding Φ fixed. Because the optimization is susceptible to local minima, we must also consider carefully how to initialize the projection and parameter matrices in this context.

The online updates for minimizing eq. (8.9) are a straightforward extension of those in the previous section. We alternatingly update the projection and parameter matrices in the following way. First, we choose an utterance $(\hat{\mathbf{x}}_n, \mathbf{y}_n)$ at random from the training corpus. Then, for each individual model, we update its parameter matrix by:

$$\Lambda^\alpha \;\leftarrow\; \Lambda^\alpha \;+\; \eta_\Lambda \frac{\partial}{\partial \Lambda^\alpha} \left[\mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{y}_n) - \mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{s}_n^{\alpha *}) \right], \tag{8.10}$$

where Λ is the factorized matrix of Φ (eq. 6.13), and the state sequence $\mathbf{s}_n^{\alpha *}$ is computed from the margin-based decoding in eq. (7.2). The right hand side of eq. (8.10) depends

on the current value of the parameter matrix $\Phi_{\mathcal{M}}$ and the projection matrix H. We optionally repeat this online update for several additional utterances. Following these updates for the model parameter matrices, we perform updates for the feature projection matrix. Given an utterance $(\hat{\mathbf{x}}_m, \mathbf{y}_m)$ at random from the training corpus, we update the projection matrix H by:

$$H \leftarrow H + \eta_H \frac{\partial}{\partial H} \sum_{\alpha} \left[\mathcal{D}^{\alpha}(\hat{\mathbf{x}}_m, \mathbf{y}_m) - \mathcal{D}^{\alpha}(\hat{\mathbf{x}}_m, \mathbf{s}_m^{\alpha *}) \right]. \tag{8.11}$$

The right hand side of eq. (8.11) depends on the current value of the projection matrix H and the parameter matrices Φ. Note that unlike the update in eq. (8.10), all models contribute to the optimization of the projection matrix H through the summation in the gradient. For more stable learning, we often perform the update in eq. (8.11) in a mini-batch fashion, averaging over small sets of training utterances. We continue this procedure, alternately updating the GMM and projection matrix parameters whenever the results from margin-based decoding do not match the target transcriptions. The scalar learning rates η_{Φ} and η_H determine the step sizes; in practice, we tune them independently to achieve the fastest convergence.

8.7 Experiments

Our experiments had two main goals: first, to test whether feature adaptation can improve phoneme recognition beyond the usual gains of discriminative training; second, to investigate the potential benefits of parameter-tying in this context. Our baseline systems were discriminatively trained HMMs with traditional cepstra, delta-cepstra, and delta-delta-cepstra as features (the results are shown in Table 7.1). Our front end computed $d_0 = 13$ mel-frequency cepstral coefficients (MFCCs) in each analysis window; initial acoustic features were computed by linearly combining the cepstra across 13 consecutive analysis windows (i.e., including six windows on each side of the current window); see eq. (8.3). To adapt the acoustic feature space, we concatenate all 169 cepstral features from these 13 windows, append a constant scalar feature of value one, and then estimate a 40x170 projection matrix, as in eq. (8.5). We experimented on acoustic models of different sizes, with 1, 2, 4, 8, 16 or 32 Gaussian mixture components per hidden state.

Since the optimization for acoustic feature adaptation is highly nonlinear, the results can be sensitive to how model parameters are initialized. We used the following scheme to obtain the positive results in this paper. First, we initialized all discriminatively trained models by their maximum likelihood counterparts. Second, we initialized all models with feature adaptation by setting the upper left block of H equal to H_0; thus, the MFCCs from different windows were initially combined by computing standard *delta* and *delta-delta* features. Third, in some experiments, we constrained the initially zero elements of the projection matrix H to remain zero; in other words, though the features were reweighted, the sparsity pattern of the projection matrix was not allowed to change during learning. This constraint led to more reliable convergence in the models without parameter-tying.

Table 8.1 compares the frame and phone error rates of acoustic models trained in different ways: by maximum likelihood (ML) estimation, by large margin (LM) training (Section 7.2), by large margin training with feature adaptation (LM+FA) but no parameter-tying, and by large margin training with feature adaptation and parameter-tying across models of different sizes (LM+FA+PT), using both sparse and full projection matrices H. All discriminatively trained models were initialized from the same ML baseline, thus starting from exactly the same performance. Also, for all these experiments, we fixed the margin-scaling parameter in eq. (8.6) as $\rho = 1$, rather than optimizing it on held-out data (which is somewhat expensive). This choice was based on the experiments in the previous section, where our results depended weakly on ρ as long as its value was greater than some small threshold.

The results in Table 8.1 show three general trends: first, that feature adaptation (LM+FA) improves performance beyond the already significant gains from large margin training (LM); second, that feature adaptation works best in conjunction with parameter-tying (LM+FA+PT) across different models; third, that the most general scheme for feature adaptation (without sparsity constraints on H) leads to the most improvement, provided that the learning is regularized in other ways. In particular, to obtain the results in the last column of Table 8.1, we not only tied the full matrix H across different models; we also employed a parameter-averaging update for the full matrix H, as described in eq. (6.5). Without both parameter-tying across models and

Table 8.1: Frame and phone error rates on the TIMIT test set for acoustic models of varying size, as obtained by maximum likelihood (ML) estimation and online updates for large margin training (LM), feature adaptation (FA), and parameter-tying (PT). See text for details. The best results in each row are shown in bold.

# of mix	Frame Error Rate (%)				
	H_0		sparse H		full H
	ML	LM	LM+FA	LM+FA+PT	LM+FA+PT
1	39.7	30.5	30.4	29.2	29.2
2	36.2	29.4	28.1	28.1	27.8
4	33.1	28.3	27.4	27.4	27.5
8	30.7	27.3	**27.4**	**26.6**	**26.4**
16	**29.5**	**27.3**	28.2	27.5	27.5
32	29.9	27.6	29.7	28.1	28.5

# of mix	Phone Error Rate (%)				
	H_0		sparse H		full H
	ML	LM	LM+FA	LM+FA+PT	LM+FA+PT
1	41.5	32.8	32.2	31.9	31.5
2	38.0	31.4	29.6	30.3	29.5
4	34.9	30.3	29.3	29.2	29.1
8	32.3	28.6	**28.8**	**27.8**	**27.7**
16	**30.8**	**28.8**	29.6	28.6	28.5
32	31.8	29.0	31.3	29.6	30.0

parameter-averaging over time, learning with full matrices H yielded worse results on both the development and test sets.

The exceptions to these trends are also revealing. For example, in the largest model with 8 Gaussian mixture components per hidden state, the frame and phone error rates are not improved by feature adaptation without parameter-tying; in fact, they are slightly worse. The worse performance may be due to overfitting and/or unreliable convergence. However, the performance in this model is improved when the feature adaptation in the front end is tied across different recognizers. The parameter-tying appears to mitigate the challenges of feature adaptation in large models. Specifically, it appears to dampen the fluctuations that arise in online learning, when updates are based on the decoding of individual training utterances. By tying the projection matrix across different model sizes, the larger model benefits from information that is accumulated

across different recognizers. Moreover, without parameter-tying, we observed that much smaller learning rates were required to obtain small but consistent improvements from acoustic feature adaptation.

Finally, we comment on convergence issues. In general, parameter-tying led to better results but not necessarily faster training: that is, roughly the same number of passes through the training data were required to converge (as measured by performance on the validation set). However, while the update rule in eq. (8.11) accumulates information across different models, we can always distribute the computation across multiple nodes, summing up the gradients from individual models as necessary. When implemented in this way, the total running time for learning is essentially equal to the individual running time of the largest model in the ensemble of recognizers. This approach exploits parallelism in a similar way as many implementations of batch training for large-scale ASR.

Fig. 8.1 compares the results from differently trained projection matrices. Specifically, we compare the projection matrix H_0 initialized by finite differencing (Fig. 8.1(a)) versus the best projection matrix learned by large margin training (Fig. 8.1(b)); that is, the one used to obtain the best result in the fifth column of Table 8.1. The difference between the two matrices is subtle but revealing. The discriminatively trained matrix incorporates more frames in its computation in contrast to the first-order derivative computation in the initial matrix H_0.

As a comparison, we also show the transformation matrix obtained from LDA on the same training data (Fig. 8.1(c)). LDA is a popular dimension reduction method in both machine learning and speech recognition. Interestingly, the transformation matrix from LDA also exhibits a pattern that suggests of differencing operations, but it gives more weight to every first MFCC, which is the energy of the frame, while ignoring the remaining coefficients. Table 8.2 shows the results from LDA transformation. We performed LDA on the 169-dimensional spliced MFCC vectors, with class information given by the phoneme transcription. The transformation resulted in a 39×169 matrix. The results from LDA are slightly better than the ML results using first- and second-order derivative features, but not as good as those from large margin training.

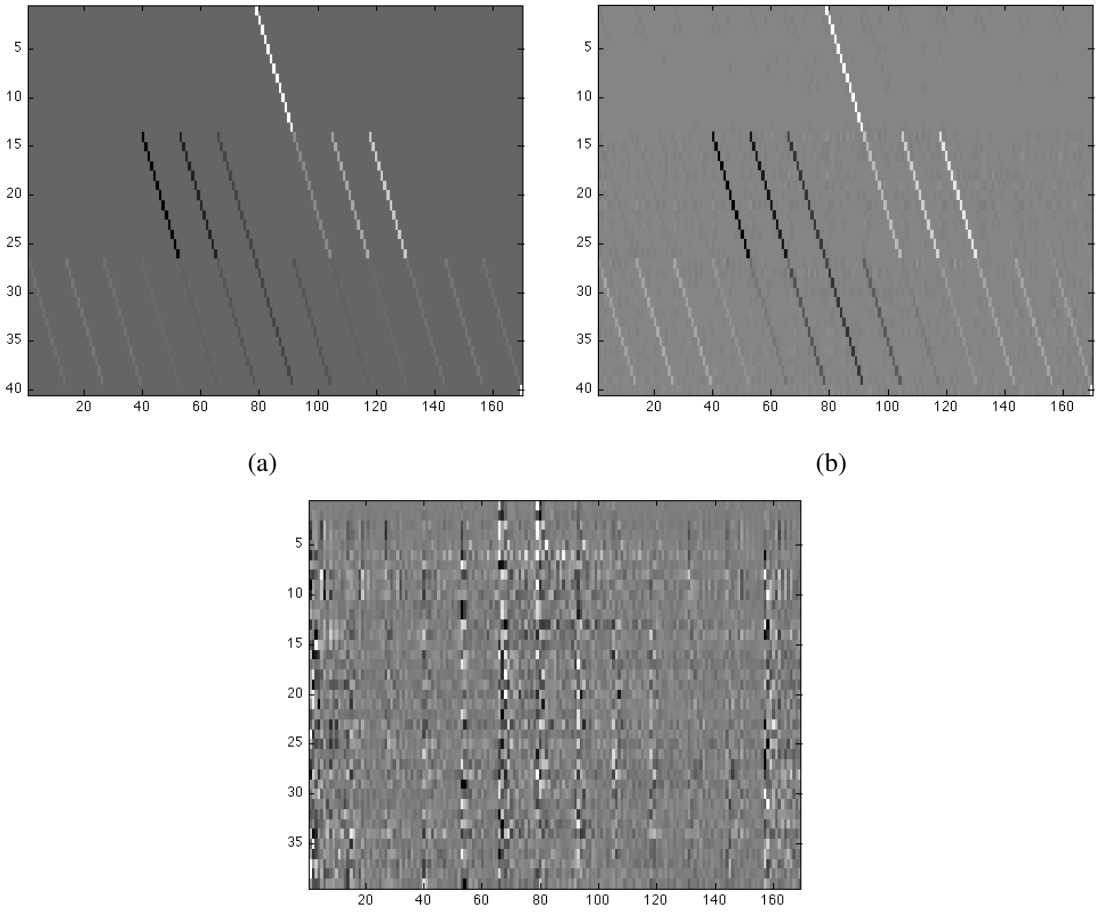

Figure 8.1: The resulted projection matrices learned by: (a) differential operations for derivative features; (b) large margin training with feature adaptation and parameter-tying; (c) linear discriminant analysis performed on the same training data.

Table 8.2: Frame and phone error rates of different acoustic feature transformations: delta and delta-delta MFCC (H_0), unconstrained discriminatively trained projection matrix (full H), and LDA transformation.

# of mix	Frame Error Rate (%)		
	H_0	full H	LDA
	ML	LM+FA+PT	ML
1	39.7	29.2	36.3
2	36.2	27.8	34.6
4	33.1	27.5	31.6
8	30.7	26.4	30.0
16	29.5	27.5	29.4
32	29.9	28.5	29.7

# of mix	Phone Error Rate (%)		
	H_0	full H	LDA
	ML	LM+FA+PT	ML
1	41.5	31.5	39.2
2	38.0	29.5	37.3
4	34.9	29.1	33.4
8	32.3	27.7	31.7
16	30.8	28.5	31.3
32	31.8	30.0	31.3

8.8 Related work

Feature space projection, or dimension reduction for acoustic features, is not new in the speech community. Transformations of the feature space have proven very successful in many tasks, such as large and small vocabulary recognition and speaker adaptation [KA98, SPGC00, THP+09, SZP01, GB01]: they are now a standard part of the feature extraction process in modern speech systems [PP02, SRP09]. One explanation of the success is that the traditional way to extract acoustic features, such as MFCCs and their derivatives, is not geared to class discrimination. The main benefit of MFCCs is that they decorrelate the spectral features from nearby parts of the frequency spectrum, usually by discrete cosine transform (DCT). However, it is not clear that how much benefit DCT has in terms of class discrimination despite it retains orthogonality in the transformed space.

Perhaps the earliest work on acoustic feature transformations applied LDA to the speech signal. LDA is a popular discriminative method for finding the projection which best separates different classes of data, while maintaining a small variance within each class [Fuk90]. LDA has shown consistent improvements in small-vocabulary speech systems but mixed results on large vocabulary ones [SPGC00, HUN92]. Following the notation in this chapter, and letting H denote a linear transformation matrix, LDA

computes the transformed feature space $\mathbf{y} = H\mathbf{x} \in R^d$ by maximizing

$$J = \frac{HBH}{HWH}, \tag{8.12}$$

where B is the covariance between classes and W is the covariance matrix within classes. The maximum of eq. (8.12) has a closed form solution given by the eigenvectors of $W^{-1}B$. In the context of dimensionality reduction, one computes the low rank projection H_d from the $d < D$ eigenvectors of $W^{-1}B$ with the largest d eigenvalues.

The original form of Fisher's LDA in eq. (8.12) is best suited when the classes of data are Gaussian distributed with identical covariance matrices (but different means). The maximum likelihood estimation of the restricted Gaussian model has been shown to be equivalent to the solution of reduced rank LDA [Cam84], with a number of prior assumptions. The first assumption is that the discriminative information only exists in the first d-dimensional subspace; second, all Gaussian clusters are of equal variances. Following the work, Hastie et al [HT96] generalized the approach to GMM-distributed class data; however, the equal covariance restriction is still strictly enforced.

The assumption of equal covariance matrices may result in an inconsistency between LDA and the acoustic modeling subsequently used in speech systems, and this might explain the non-optimal performance in those preliminary experiments. In order to compute the relevant acoustic features with the most discriminative information and also suited to the subsequent acoustic models, the equal covariance assumption has to be relaxed. Heteroscedastic linear discriminant analysis (HLDA) [KA98] extends LDA by dropping the equal covariance constraint among different classes. It has proven widely useful in speech recognition [SPGC00, OHJ03, ZM05] and has been adopted as a standard step in acoustic feature processing [PP02, SRP09].

Kumar et al [KA98] initiated this line of work with a maximum likelihood-based algorithm to optimize the feature space transformation along with the model parameters. The approach was motivated by Campbell [Cam84], and released the equal covariance constraints by modelling each class with a different covariance matrix. Given the data of each class can be modeled by single Gaussian distribution, the log-likelihood L_{HLDA}

of the training data is

$$L_{HLDA} = \sum_{i=1}^{N} -\frac{1}{2}\{(H^T x_i - \mu_{y_i})^T \Sigma_{y_i}^{-1}(H^T x_i - \mu_{y_i}) + \log((2\pi)^D |\Sigma_{y_i}|)\} + \log|H|, \tag{8.13}$$

where H is a $D \times D$ full rank matrix. By assuming that only the first d dimensions of the transformed space contain the class discrimination information, the means μ_j, covariance matrices Σ_j, and the transformation matrix H can be partitioned into two parts: one for the informative d components, and one for the globally invariant $D - d$ components. A joint optimization of all parameters is infeasible for eq. (8.13); however, the task can be simplified by first calculating the optimal values of one parameter while fixing the others. With a fixed transformation matrix, the optimization of the means and covariance matrices can be done by the standard maximum likelihood estimation. By differentiating eq. (8.13) with respect to means and covariance matrices, there are closed form solutions for these parameters:

$$\hat{\mu}_j^d = H_d^T \bar{x}_j \tag{8.14}$$

$$\hat{\mu}^{D-d} = H_{D-d}^T \bar{x} \tag{8.15}$$

$$\hat{\Sigma}_j^d = \frac{1}{N_j}(H_d^T W_j H_d) \tag{8.16}$$

$$\hat{\Sigma}^{D-d} = \frac{1}{N}(H_{D-d}^T B H_{D-d}). \tag{8.17}$$

Substituting $\hat{\mu}_j$ and $\hat{\Sigma}_j$ back into eq. (8.13), one can obtain a new likelihood function in terms of H, which can be maximized by

$$\hat{H} = \text{argmax}_H\{-\frac{N}{2}\log|H_{D-d}^T \bar{B} H_{D-d}| - \sum_j -\frac{N_j}{2}\log|H_d^T \bar{W}_j H_d| + N\log|H|\}, \tag{8.18}$$

where $\bar{B}$ and $\bar{W}_j$ are the estimated between- and within-class covariance matrices with respect to $\hat{\mu}$. Although the optimization of both model parameters and feature transformation are under the same objective function, the training of HLDA has to take place in separate procedures. That is, the model parameters are obtained by EM algorithm while fixing the transformation matrix, and the transformation parameter is learned by a gradient method while fixing the Gaussian model parameters.

There are several alternative HLDA methods to learn the discriminative projection. Omar et al [OHJ03] proposed to use a discriminative objection function instead

of ML criterion. The objective functions in ML-based HLDA projections aim at maximizing the distance between clusters at the reduced feature space. The authors argued that the mutual information between the projected feature space and the set of clusters might be a better objective for recognition. Similarly, Zhang et al [ZM05] proposed to use minimum phone error (MPE) as the criterion to preserve during the projection.

Although ML-based and discriminative criterion-based methods incorporate the feature projection in the objective functions, they do not learn feature transformation parameters *along with* model parameters under the same optimization. To see this point, for example in [KA98], they performed an EM algorithm for the model parameters, one gradient-based search for the transformation parameters, and one more search for the model parameters. Our framework distinguishes from others in the sense that it learns both model-space and feature-space parameters simultaneously, until it meets the convergence criterion.

More recently, neighbourhood component analysis (NCA) [GRHS05] has been applied to speech recognition tasks. The goal of NCA is to find a distance metric that maximizes the performance of k-nearest neighbour (k-NN) classification, where the distance measure is defined as $\text{Dist}_{NCA} = (A\mathbf{x} - A\mathbf{y})^T(A\mathbf{x} - A\mathbf{y})$. If A is a low-rank matrix, finding the optimal distance metric A^TA is equivalent to finding the optimal dimension reduction. Inspired by NCA, Singh-Miller et al [SMCH07] proposed to learn the acoustic feature transformation to minimize the k-NN classification loss in the transformed space. Although NCA is a discriminative method, it is not working directly toward minimizing the loss measure for decision making in speech recognition. Therefore, as shown in [SMCH07], the classification results are more promising than their recognition results. Another potential difficulty of NCA is its ability to scale to large datasets due to the fact that its complexity grows quadratically with the number of training data, and a parallelized computation is usually implemented.

Acoustic feature transformation is also widely used for rapid speaker adaptation. Although addressing different problems than acoustic feature projection (i.e., dimensionality reduction), the optimization of speaker adaptation problems requires solving similar equations as the one described in HLDA [LW95, Gal98b, SZP01]. The success of a speech model relies on a sophisticated training with a large amount of training

data to cover the possible variability in the real environment. However, even the current speaker-independent models have achieved a good performance, speaker-dependent systems in general still yield an even better performance [LW95]. Yet the training of speaker-dependent systems require a significant amount of data from each individual speaker, which is impractical in most circumstances. Therefore, the goal of speaker adaptation is to adapt a well-trained speaker-independent model to a speaker-dependent model at testing phase by a small amount of speaker-specific data. The adaptation is carried out by finding a transformation on the model parameters or on the feature space such that the objective function is optimized given the speaker dependent training data.

Specially in the constrained maximum likelihood linear regression (MLLR), a linear transform is applied to the feature vectors and learned by maximizing the likelihood of the acoustic data [Gal98b]. The objective function is identical to eq. (8.13), but with several differences. The first difference is the full-rank transformation matrix H in MLLR, which makes an efficient iterative algorithm based on forward-backward probability feasible. Second, there is usually a very small amount of speaker-dependent data available for training, and thus how to adapt all phone models with the limited data is a practical issue. The training utterances may not contain enough data for every phone model. A tying between similar states is usually committed to share the training data [LW95]. This is analogous to the tying approach for training context-dependent models [YOW94, DM94], but with flexibility regarding the available data in the utterances (e.g. how/which states to tie).

Saon et al [SZP01] has noticed the parallelism between HLDA and MLLR, and developed a speaker-dependent feature projection based on MLLR training for speaker adaptation problems. This is a new merge of the front-end training phase and the back-end testing phase, since conventionally the projection is done at the front-end feature processing step, with training data, and the subsequent training, testing and adaptation processes are all performed on the projected space. Supposedly the unknown testing data may have a different projected space that contains the maximal discrimination information, and it is beneficial to capture this variation in addition to simply rotating the feature space.

In light of all the above discussion, the acoustic feature adaptation proposed in

this chapter is potentially applicable and relevant to speaker adaptation problems. There are several insights. First, for the problem of a small amount of data, the parameter tying exploited in Section 8.5 can serve as a base. Instead of tying between recognizers, the tying can be performed among similar states in different phone models, as determined by the available data. Second, the rank of the transformation matrix in our framework is flexible. In contrast to HLDA, there is no need to learn a global projection in the rejected space, and thus the training could be easier with our framework. A recent work of one of our counterparts has shown a promising result in this direction [LSN10].

8.9 Summary

In this chapter we have explored how to optimize the acoustic features computed by front ends for ASR. Extending the framework in previous chapters for large-margin training of CD-HMMs, we showed that standard acoustic features could be discriminatively reweighted to improve performance. Our best results were obtained by tying the feature reweighting parameters across multiple recognizers and training these different recognizers in an integrated manner. The parameter-tying across models was used to average the strongly biased gradients from individual training utterances in online learning.

Chapter 8, in part, is a reprint of the material as it appear in Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-09) 2009. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE, 2009. Chapter 5-8, in part, is a reprint of the material as it appear in IEEE Journal of Selected Topics in Signal Processing 2010. Chih-Chieh Cheng; Fei Sha; Lawrence K. Saul, IEEE Signal Processing Society, 2010. The dissertation/thesis author was the primary investigator and author of these papers.

Chapter 9

Discussion

In this thesis, we have explored the potential of online updates for discriminative training of HMMs. We have also shown that such updates can be used for large margin training and acoustic feature adaptation, yielding further improvements in performance. We conclude by summarizing our main contributions.

In Chapter 6, we identified two reparameterizations of GMMs that lead to more effective online updates. The reparameterization in eq. (6.10) aggregates the mean and covariance matrix parameters of GMMs, yielding simpler gradients and eliminating the need for multiple different learning rates. The further reparameterization in eq. (6.13) leads to significantly faster convergence. Based on extensive experimental results, we identified these reparameterizations, as well as the averaging in eq. (6.5) as best practices for discriminative training of HMMs using online updates. Moreover, as shown in Table 6.1, acoustic models trained in this way performed better than acoustic models trained by popular batch approaches such as CML and MCE.

In Chapter 7, we showed how to extend these online updates to incorporate large margin constraints. Large margin training led to further improvements in frame and phone error rates. Though the online updates for large margin training did not match the performance of previous batch implementations, they were simpler to implement and required fewer passes through the set of training utterances.

Finally, in Chapter 8, we used online updates for end-to-end training of speech recognizers. In particular, we experimented with alternating updates that jointly adapted the features computed by the front end in conjunction with large margin training of

HMMs. For our best results, we developed a novel parameter-tying scheme that tied the feature projection matrices of multiple different recognizers trained in parallel. This scheme yielded further improvements in frame and phone error rates across all model sizes; the results also surpassed those from previous batch implementations of large margin training (without acoustic feature adaptation). Though acoustic features could in principle be adapted within a batch framework, the required implementation and experimentation would be much more unwieldy.

One potential work for future direction is to apply this framework on large vocabulary continuous speech recognition (LVCSR), since there exists a great number of challenges in the problems of such a large scale, including the size of the models, the size of the training/testing data, and the unlimited length of each data. As discussed in Chapter 4, the LVCSR systems in fact employs a highly customized framework devoted to optimize the performance of the tasks. We briefly discuss how to extend our framework to LVCSR systems in the following.

LVCSR systems usually model each phoneme by a context-dependent HMM with three (or more) hidden states. These phone HMMs are concatenated to form word HMMs, sentence HMMs, and so on. Parameter estimation depends on running a forward-backward algorithm on a word lattice. Decoding is based on a two-pass decoder, with the first pass generating the lattice and the second pass performing a constrained search using a high-order N-gram language model [GY08]. This implementation raises challenges that do not appear in our framework, particularly the training data with partial transcriptions and the decoding of a large word graph.

In our framework (see Chapter 6), we compare the statistics of the correct state sequence $\mathbf{s}$ to the most likely competing sequence $\mathbf{s}^*$. For the statistics of the correct sequence, we need the correct transcription as provided by a frame-level phonetic transcriptions. However, such transcriptions are not available for the large corpora used in LVCSR. These corpora are transcribed at the *word-level*; the word-level transcriptions do not provide frame-by-frame phonetic alignments.

The statistics from the correct sequence are also required in the discriminative training of most LVCSR systems (e.g. for the *numerator* of the objective function in MMI training). To obtain these statistics, a reference HMM for the correct sequence is

built by concatenating phone HMMs consistent with the word-level transcriptions. The frame-level phone labels are then obtained by a forced alignment of the utterance by the reference HMM. The parameters of the reference HMM are usually initialized by a previously built recognizer, and the reference HMM itself is usually represented by a lattice.

When extending our framework to LVCSR systems, the same alignment is required to obtain frame-level transcriptions. In our case, we can perform the forced-alignment once at the beginning or each update. We explored this idea in section 7.3.3, where we used forced alignment to derive phonetic boundaries, not assuming they were available from the transcription. Another idea is to perform the forced alignments less frequently - say, once every several updates - as is done in *direct loss minimization* [MHK10]. We believe either approach is viable for LVCSR implementation.

For discriminative training, we also need statistics from competing hypotheses; these statistics are more expensive to compute. In LVCSR implementations of MMI [VOWY97], a subset of competing hypotheses are encoded by a word lattice, and a constrained search estimates the occupation probabilities by a forward-backward algorithm. The lattice makes it possible to compute the statistics of the N-best hypotheses as opposed to just the strongest competitor (as in Perceptron-like training). We can use a lattice in the same way for the large margin HMMs in this thesis. In particular, the derivatives of the discriminant function $\mathcal{D}(\mathbf{x}, \mathbf{s}^*)$ can be expressed in terms of the posterior occupation probabilities (Appendix A). For one-best list, the required statistics can be computed by a forward pass of decoding and a backward pass along the decoded path. For N-best lists, the state occupation probabilities can be computed by forward-backward algorithms that average over all paths encoded by the lattice.

A final concern for LVCSR is how to incorporate the large margin constraints. Recently, a "boosted" variant of MMI has been proposed [PKK$^+$08] which can be viewed as enforcing a margin on competing sequences. In this approach, the margin constraints were incorporated into lattice calculations by penalizing wrong alignments on the fly. We can use a similar strategy for the margin constraints in our approach. However, one question for LVCSR is whether the margin constraints are best formulated in terms of frame-level Hamming distances. For ASR, the misalignment between

two transcriptions is more appropriately measured by the word-level edit distance than the frame-level Hamming distance. However, edit distance are more expensive to compute, and it is unclear how to incorporate them into lattice computations for our framework. One possible approximation is the "phone accuracy" used in minimum phone error (MPE) criterion [PW02]; this possibility remains a direction for future work.

Another interesting direction is to apply the ideas in Chapter 8 to the problem of speaker adaptation. Speaker adaptation has proved to improve the speech recognition systems by a significant amount, and has become part of the standard process in the state-of-the-art speech systems [GY08]. Inspired by a recently-developed line research [LSN10, HLSN08], the incorporation of our framework for speaker adaptive training is practicable, building on the basis of the acoustic feature adaptation introduced in Chapter 8.

Finally, we conclude by mentioning the possibility of training on unlimited data - for example, if the model is updated daily with renewed training data [BDG$^+$09]. Online learning has shown success in this regime for other applications [MSSV09]. Indeed, this regime is precisely where one expects the biggest payoff from online methods, as we have considered in this thesis.

Appendix A

Derivation of the gradients

The online update (eq.(6.4)) is an instance of generalized stochastic gradient methods. The major computation in the update is dominated by the gradient computation according to different parameterizations. In this work, we have described updates in Φ (eq. (6.12)), in the factorized parameters Λ (eq. (6.14)), and in the projection matrix H (eq. (8.11)). Here we show the derivation of the gradient with respect to Λ as an example. Gradients with the other parameterizations can be derived in a similar way.

Recall that the loss function (eq.(6.3)) is defined as

$$\mathcal{L}(\Lambda) \;=\; \sum_{n} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) - \mathcal{D}(\mathbf{x}_n, \mathbf{y}_n)\right]^+, \tag{A.1}$$

where $[z]^+ = \max(z, 0)$ indicates the nonnegative hinge function and $\mathbf{s}_n^*$ is the Viterbi decoding sequence. The order of the two terms inside the hinge function does not matter too much - minimizing a loss function is equivalent to maximizing the negative loss function - as long as the update is verified using either a gradient *descent* or a gradient *ascent* method.

For simplification, we assume the HMM uses single Gaussian as its emission density function. Thus the discriminant of an observation sequence $\mathbf{x}$ and a state transition sequence $\mathbf{s}$ (eq. (6.1)) can be written as

$$\mathcal{D}(\mathbf{x}, \mathbf{s}) = \log \mathcal{P}(s_1) + \sum_{t>1} \log \mathcal{P}(s_t|s_{t-1}) - \frac{1}{2} \sum_{t} x_t^T \Phi_{s_t} x_t, \tag{A.2}$$

in which we parameterize the Gaussian density function by Φ_{s_t} matrices (Section 6.4). By taking gradient on $\mathcal{D}(\mathbf{x}, \mathbf{s})$ with respect to the parameter of the kth state Φ^k, we

obtain the equation

$$\frac{\partial \mathcal{D}}{\partial \Phi^k} = -\frac{1}{2}\sum_t \delta_{s_t,k}(x_t x_t^T), \tag{A.3}$$

where $\delta_{l,r}$ is an indicator function with a value 1 if $l = r$.

For a faster convergence by an unconstrained update, we need to work on the square root matrix Λ (Section 6.5), where

$$\Phi = \Lambda\Lambda^T. \tag{A.4}$$

Combining eqs. (A.3) and (A.4), the gradient on Λ is decomposed and derived by chain rule:

$$\begin{aligned}
\frac{\partial \mathcal{D}}{\partial \Lambda^k_{lm}} &= \sum_{ij} \frac{\partial \mathcal{D}}{\partial \Phi^k_{ij}} \frac{\partial \Phi^k_{ij}}{\partial \Lambda^k_{lm}} \\
&= \sum_{ij} -\frac{1}{2}\sum_t (x_t x_t^T)_{ij}\delta_{s_t,k}(\delta_{il}\Lambda^k_{jm} + \delta_{jl}\Lambda^k_{im}) \\
&= -\frac{1}{2}\sum_t \delta_{s_t,k}\sum_{ij}(x_t x_t^T)_{ij}(\delta_{il}\Lambda^k_{jm} + \delta_{jl}\Lambda^k_{im}) \\
&= -\frac{1}{2}\sum_t \delta_{s_t,k}(\sum_j (x_t x_t^T)_{lj}\Lambda^k_{jm} + \sum_i (x_t x_t^T)_{il}\Lambda^k_{im}) \\
&= -\frac{1}{2}\sum_t \delta_{s_t,k}(2(x_t x_t^T)_{l(\boldsymbol{.})}\Lambda^k_{(\boldsymbol{.})m}),
\end{aligned}$$

where $(\boldsymbol{.})$ denotes all elements in the row or column. We can see that the gradient on the lmth element in Λ^k is simply the dot product of the lth row in $x_t x_t^T$ and the mth column in Λ^k. Hence, we obtain the gradient on the matrix Λ^k by

$$\frac{\partial \mathcal{D}}{\partial \Lambda^k} = -\sum_t \delta_{s_t,k}(x_t x_t^T)\Lambda^k. \tag{A.5}$$

In the context of online learning, we only see one observation at a time and therefore we only need to calculate the gradient of the loss regarding the nth observation $\mathcal{L}_n$:

$$\frac{\partial \mathcal{L}_n}{\partial \Lambda^k} = -\text{step}[\mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) - \mathcal{D}(\mathbf{x}_n, \mathbf{y}_n)]\sum_t (\delta_{s_t^*,k} - \delta_{y_t,k})(x_t x_t^T)\Lambda^k. \tag{A.6}$$

Finally, the update of the parameters accordingly follows those described in Section 6.5.

It is straightforward to apply the gradient derivation to HMMs with Gaussian mixture models. Instead of having the $\{0, 1\}$ indicator, every outer product $x_t x_t^T$ is weighted by the posterior probability $p(km|x_t)$, where m is the index of Gaussian mixture component in the state k. Intuitively, the gradient can be viewed as a (weighted) count of occurrences of state k (or Gaussian mixture component km) in the transcription and the Viterbi decoding sequence. This is analogous to the Perceptron training on discrete HMMs in [Col02], where they increase/decrease the feature weights based on the occurrences of features in the transcription and the predicted sequence. However, unlike their work, we work on continuous density functions, and the updates are thus based on a sum of outer products of the examples in the sequences.

In our implementation, we launch the training and testing processes with MATLAB interfaces, and the major computation is written in C language with the Basic Linear Algebra Subprograms (BLAS) library. The computation in each "pass" of training data is almost the same as that of the batch counterpart of this work [SS09], except that we need to do Viterbi decoding for every upcoming sequence and therefore add up some overhead. Nevertheless, we can have thousands of updates done in one single pass of training data in contrast to one single update for one pass in batch learning algorithms.

Bibliography

[ASS00] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1:113–141, April 2000.

[ATH03] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov Support Vector Machines. In *Proceedings of the International Conference on Machine Learning (ICML-03)*, pages 3 – 10, 2003.

[BB08] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. MIT Press, 2008.

[BBdSM86] 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 International Conference of Acoustic, Speech and Signal Processing (ICASSP-86)*, pages 49–52, Tokyo, 1986.

[BDG+09] Janet M Baker, Li Deng, James Glass, Sanjeev Khudanpur, Chin-hui Lee, Nelson Morgan, and Douglas O'Shaughnessy. Developments and directions in speech recognition and understanding, Part 1. *IEEE Signal Processing Magazine*, 26(3):75–80, 2009.

[BGV92] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[Bis96] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1996.

[BL04] Léon Bottou and Yann LeCun. Large scale online learning. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[BLB97] Léon Bottou, Yann Le Cun, and Yoshua Bengio. Global training of document processing systems using graph transformer networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-97)*, pages 489–493, Puerto Rico, 1997.

[Bn98] Tom Brø ndsted. A SPE based distinctive feature composition of the CMU Label Set in the TIMIT database. In *Techinical Report IR 98-1001*. Center for PersonKommunikation, Aalborg University, 1998.

[Bot04] Léon Bottou. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

[Bot07] Léon Bottou. Lerning with large datasets. In *NIPS Tutorials*. 2007.

[Bur98] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[BVG04] Sreeram Balakrishnan, Karthik Visweswariah, and Vaibhava Goe. Stochastic gradient adaptation of front-end parameters. In *Proceedings of Interspeech-2004*, pages 1–4, 2004.

[Cam84] N. A. Campbell. CANONICAL VARIATE ANALYSISA GENERAL MODEL FORMULATION. *Australian Journal of Statistics*, 26:86–96, 1984.

[CDSSS03] Koby Crammer, Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. In *NIPS*, 2003.

[CH68] Noam Chomsky and Morris Halle. *The Sound Pattern of English*. Harper and Row, 1968.

[CKS03] Koby Crammer, J Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems 16*, 2003.

[Col02] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-02)*, volume 10, pages 1–8, 2002.

[Cov65] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.

[Cra10] Koby Crammer. Efficient online learning with individual learning-rates for phoneme sequence recognition. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP-10)*, pages 4878–4881, 2010.

[CS00] Koby Crammer and Yoram Singer. On the Learnability and Design of Output Codes for Multiclass Problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory (COLT-00)*, pages 35–46, 2000.

[CS01] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, 2:265–292, 2001.

[CS03] Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:2003, 2003.

[CSS09a] C. C. Cheng, F. Sha, and L. K. Saul. Acoustic feature adaptation in large margin hidden Markov models. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-09)*, pages 87–92, Merano, Italy, 2009.

[CSS09b] C. C. Cheng, F. Sha, and L. K. Saul. A fast online algorithm for large margin training of continuous density hidden markov models. In *Proceedings of Interspeech-2009*, pages 668–671. 2009.

[CSS09c] C. C. Cheng, F. Sha, and L. K. Saul. Matrix updates for perceptron training of continuous density hidden markov models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 153–160. 2009.

[CV95] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[DCP08] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-Weighted Linear Classification. In *Proceedings of the 25th International Conference on Machine Learning*, pages 264–271, 2008.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[DM94] Vassilios Digalakis and Hy Murveit. High-accuracy large-vocabulary speech recognition using mixture tying and consistency modeling. In *Proceedings of the workshop on Human Language Technology (HLT-94)*, pages 313–318. Association for Computational Linguistics, 1994.

[DRrMH10] George Dahl, Marc'Aurelio Ranzato, Abdel rahman Mohamed, and Geoffrey Hinton. Phone recognition with the mean-covariance restricted boltzmann machine. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 469–477. 2010.

[DSsS08] Ofer Dekel, Shai Shalev-shwartz, and Yoram Singer. The forgetron: a kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

[DYA06] Li Deng, Dong Yu, and Alex Acero. Structured speech modeling. *IEEE Transactions on Audio, Speech & Language Processing*, 14(5):1492–1504, 2006.

[Efr75] Bradley Efron. The Efficiency of Logistic Regression Compared to Normal Discriminant Analysis. *Journal of the American Statistical Association*, 70(352):892– 898, 1975.

[FS99] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Machine Learning*, pages 277–296, 1999.

[FSSSU06] Michael Fink, Shai Shalev-Shwartz, Yoram Singer, and Shimon Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 313–320, New York, New York, USA, 2006. ACM Press.

[Fuk90] K. Fukanaga. *Introduction to statistical pattern recognition*. Academic Press, 1990.

[Gal98a] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12:75–98, 1998.

[Gal98b] Mark J. F. Gales. Maximum Likelihood Linear Transformations for HMM-based Speech Recognition. *Computer Speech and Language*, 12:75–98, 1998.

[Gal99] Mark J. F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, May 1999.

[GB01] Asela Gunawardana and William Byrne. Discriminative speaker adaptation with conditional maximum likelihood linear regression. In *Proceedings of the 7th European Conf. on Speech Communication and Technology (Eurospeech-2001)*, volume 01pp, pages 1203–1206, 2001.

[Gen02] Claudio Gentile. A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res.*, 2:213–242, 2002.

[GKNN91] P.S. Gopalakrishnan, D. Kanevsky, a. Nadas, and D. Nahamoo. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory*, 37(1):107–113, 1991.

[GMAP05] Asela Gunawardana, Miland Mahajan, Alex Acero, and John C. Platt. Hidden conditional random fields for phone classification. In *Proceedings of Nineth European Conference on Speech Communication and Technology (EuroSpeech 2005)*, pages 1117–1120, Lisbon, 2005.

[GRHS05] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 513–520, Cambridge, MA, 2005. MIT Press.

[GY08] Mark J. F. Gales and Steve Young. The Application of Hidden Markov Models in Speech Recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2008.

[HAH01] X. Huang, A. Acero, and H.-W. Hon. *Spoken language processing*. Prentice-Hall, 2001.

[Her90] Hynek Hermansky. Perceptrual Linear Predictive (PLP) Analysis of Speech. *J. Acoustic Soc. America*, 87(4):1738–1752, 1990.

[HLSN08] Georg Heigold, Patrick Lehnen, Ralf Schluter, and Hermann Ney. ON THE EQUIVALENCE OF GAUSSIAN AND LOG-LINEAR HMMS. In *INTERSPEECH-2008*, pages 273–276, 2008.

[HT96] Trevor Hastie and Robert Tibshirani. Discriminant Analysis by Gaussian Mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):155–176, 1996.

[HUN92] R. Haeb-Umbach and Hermann Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92)*, volume 1, pages 13–16, 1992.

[HW91] Patrick Haffner and Alex Waibel. Multi-state time delay neural networks for continuous speech recognition. In *Advances in Neural Information Processing Systems 4*, pages 135–142, 1991.

[JFY09] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, May 2009.

[JH98] Tommi S. Jaakkola and David Haussler. Exploiting Generative Models in Discriminative Classifiers. In *Advances in Neural Information Processing Systems 11*, 1998.

[JK92] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. Sig. Proceedings*, 40(12):3043–3054, 1992.

[JLL06a] H. Jiang, X. Li, and C.J. Liu. Large margin hidden markov models for speech recognition. *IEEE Trans. on Audio, Speech and Language Processing*, 14(5):1584–1595, 2006.

[JLL06b] Hui Jiang, Xinwei Li, and Chaojun Liu. Large Margin Hidden Markov Models for Speech Recognition. *IEEE Trans. on Audio, Speech and Language Processing*, 14(5):1584–1595, 2006.

[Joa99] Thorsten Joachims. *Making large-scale support vector machine learning practical*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[JR05] Biing-Hwang Juang and Lawrence R. Rabiner. *Automatic Speech Recognition A Brief History of the Technology Development*. 2005.

[KA98] Nagendra Kumar and Andreas G. Andreou. Heteroscedastic discriminant analysis and reduced rank hmms for improved speech recognition. *Speech Communication*, 26(4):283–297, 1998.

[KM01] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[KSSB+06] J. Keshet, S. Shalev-Shwartz, S. Bengio, Y. Singer, and D. Chazan. Discriminative kernel-based phoneme sequence recognition. In *Proceedings of Interspeech-06*, pages 1284–1287, 2006.

[KSSSC07] Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, and Dan Chazan. A Large Margin Algorithm for Speech-to-Phoneme and Music-to-Score Alignment. *IEEE Trans. Audio, Speech, and Language Processing*, 15(8):2373–2382, November 2007.

[KSST08] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 440–447, 2008.

[KSW01] Jyrki Kivinen, Alex J. Smola, and Robert C Williamson. Online Learning with Kernels. In *Advances in Neural Information Processing Systems 14*, 2001.

[KVY93] S. Kapadia, V. Valtchev, and S.J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93)*, pages 491–494. Ieee, 1993.

[LBOM98] Yann LeCun, Leon Bottou, Geneviere B. Orr, and Klaus-Robert Muller. *Efficient BackProp*. Springer, 1998.

[LH88] K. F. Lee and H. W. Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 37:1641–1648, 1988.

[LJ08] P. Liang and M. I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *International Conference on Machine Learning (ICML)*, pages 584–591, 2008.

[LKS86] L. F. Lamel, R. H. Kassel, and S. Seneff. Speech database development: design and analysis of the acoustic-phonetic corpus. In L. S. Baumann, editor, *Proceedings of the DARPA Speech Recognition Workshop*, pages 100–109, 1986.

[LL02] Yi Li and Philip M. Long. The Relaxed Online Maximum Margin Algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

[LM05] Jonathan Le Roux and Erik McDermott. Optimization methods for discriminative training. In *Proceedings of Nineth European Conference on Speech Communication and Technology (EuroSpeech-05)*, pages 3341–3344, 2005.

[LSN10] Jonas Lf, Ralf Schlter, and Hermann Ney. Discriminative adaptation for log-linear acoustic models. In *Proceedings of INTERSPEECH-2010*, pages 1648–1651. 2010.

[LW95] C. J. Leggetter and Philip C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, 9(2):171–185, 1995.

[LW01] Kyung-tak Lee and Christian J Wellekens. Dynamic Sharings of Gaussian Densities using Phonetic Features. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-01)*, pages 425–428, 2001.

[LYL07] J. Li, M. Yuan, and C.H. Lee. Approximate test risk bound minimization through soft margin estimation. *IEEE Trans. on Speech, Audio and Language Processing*, 15(8):2392–2404, 2007.

[MCP05] Ryan Mcdonald, Koby Crammer, and Fernando Pereira. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, 2005.

[Mer88] B. Merialdo. Phonetic recognition using hidden Markov models and maximum mutual information training. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-88)*, pages 111–114, 1988.

[MH03] Naren Malayath and Hynek Hermansky. Data-driven spectral basis functions for automatic speech recognition. *Speech Communication*, 40(4):449–466, 2003.

[MHK10] David McAllester, Tamir Hazan, and Joseph Keshet. Direct loss minimization for structured prediction. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1594–1602. 2010.

[MHR+07] E. McDermott, T. Hazen, J. L. Roux, A. Nakamura, and S. Katagiri. Discriminative training for large-vocabulary speech recognition using minimum classification error. In *IEEE Transactions on Audio Speech and Language Processing*. 2007.

[MP06] Ryan McDonald and Fernando Pereira. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, 2006.

[MSSV09] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the International Conference on Machine Learning (ICML-09)*, pages 681–688. 2009.

[MWS08] John McDonough, Matthias Wlfelc, and Emilian Stoimenov. On maximum mutual information speaker-adapted training. *Computer Speech and Language*, 22(2):130–147, 2008.

[Nad83] A. Nadas. A decision-theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31(4):814–817, 1983.

[NJ02] 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*, volume 14, pages 841–848. 2002.

[OHJ03] Mohamed Kamal Omar and Mark Hasegawa-Johnson. Maximum Conditional Mutual Information Projection For Speech Recognition. In *Proceedings of the 8th European Conference on Speech Communicaiton and Technology (EUROSPEECH)*, pages 505–508, 2003.

[OKC08] Francesco Orabona, Joseph Keshet, and Barbara Caputo. The Projectron : a Bounded Kernel-Based Perceptron. In *Proceedings of the 25th International Conference on Machine Learning*, pages 720–727, 2008.

[PCSt99] John C Platt, Nello Cristianini, and John Shawe-taylor. Large Margin DAGs for Multiclass Classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 1999.

[PKK+08] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariahy. Boosted MMI for model and feature-space discriminative training. In *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-08)*, pages 4057–4060, 2008.

[Pla99] John C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[PP02] M. Padmanabhan and M. Picheny. Large-vocabulary speech recognition algorithms. *Computer*, 35(3):42–50, March 2002.

[PPK07] Slav Petrov, Adam Pauls, and Dan Klein. Learning structured models for phone recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 897–905, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[PW02] D. Povey and P.C. Woodland. Minimum phone error and i-smoothing for improved discriminative training. In *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-02)*, pages 105–108, Orlando, FL, 2002.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.

[RJ93] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[RK04] Ryan Rifkin and Aldebaro Klautau. In Defense of One-Vs-All Classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[SB00] Alexander J. Smola and Peter J. Bartlett, editors. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, USA, 2000.

[SBV95] Bernhard Schölkopf, Christopher J.C. Burges, and Vladimir N. Vapnik. Extracting support data for a given task. In *Proceedings of the first International Conference on Knowledge Discovery & Data Mining (KDD-95)*, 1995.

[SEK+00] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky. Feature extraction using non-linear transformation for robust speech recognition on the aurora database. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 2:II1117–II1120, 2000.

[SFK+05] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman. MLLR transforms as features in speaker recognition. In *Proceedings of INTERSPEECH-2005*, pages 2425–2428, 2005.

[SMCH07] N. Singh-Miller, M. Collins, and T. J. Hazen. Dimensionality reduction for speech recognition using neighborhood components analysis. In *Proceedings of Interspeech-07*, pages 1158–1161, 2007.

[SNE+10] G.S.V.S. Sivaram, Sridhar Krishna Nemala, Mounya Elhilali, Trac D. Tran, and Hynek Hermansky. Sparse coding for speech recognition. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP-10)*, pages 4346–4349, 2010.

[SP08] G. Saon and D. Povey. Penalty function maximization for large margin HMM training. In *Proceedings of Interspeech-2008*, pages 920–923, 2008.

[SPGC00] George Saon, Mukund Padmanabhan, Ramesh Gopinath, and Scott Chen. Maximum Likelihood Discriminant Feature Spaces. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP-00)*, volume 2, pages 1129–1132, 2000.

[SRP09] Tara N Sainath, Bhuvana Ramabhadran, and Michael Picheny. An exploration of large vocabulary tools for small vocabulary phonetic recognition. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU-09)*, pages 359–364, 2009.

[SS99] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

[SS06] Fei Sha and Lawrence K. Saul. Large margin gaussian mixture modeling for phonetic classification and recognition. In *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-06)*, 2006.

[SS07a] Fei Sha and Lawrence K. Saul. Comparison of large margin training to other discriminative methods for phonetic recognition by hidden Markov models. In *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-07)*, pages 313–316, Honolulu, HI, 2007.

[SS07b] Fei Sha and Lawrence K. Saul. Large margin hidden Markov models for automatic speech recognition. In B. Schölkopf, J.C. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 1249–1256, Cambridge, MA, 2007. MIT Press.

[SS09] F. Sha and L. K. Saul. Large margin training of continuous density hidden markov models. In J. Keshet and S. Bengio, editors, *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, pages 101–114. Wiley-Blackwell, 2009.

[SSL07] S. M. Siniscalchi, P. Schwarz, and C.-H. Lee. High-accuracy phone recognition by combining high-performance lattice generation and knowledge based rescoring. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing (ICASSP-07)*, volume 4, pages 869–872, 2007.

[SSSS07] Shai Shalev-Schwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In Zoubin Ghahramani, editor, *Proceedings of 24th International Conference on Machine Learning (ICML 2007)*, pages 807–814, Corvallis, OR, 2007. Omnipress.

[SWH+89] H. Sawai, A. Waibel, P. Haffner, M. Miyatake, and K. Shikano. Parallelism, Hierarchy, Scaling in Time-Delay Neural Networks for Spotting Japanese Phonemes/CV-Syllables. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 81–88, Washington D.C., 1989. IEEE, New York.

[SZP01] George Saon, G. Zweig, and M. Padmanabhan. Linear feature space projections for speaker adaptation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (ICASSP-01)*, pages 325–328, 2001.

[TGK04] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[THJA04] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML-04)*, pages 104–112, Banff, Canada, 2004.

[THP+09] Muhammad Ali Tahir, Georg Heigold, Christian Plahl, Ralf Schl, and Hermann Ney. Log-Linear Framework for Linear Feature Transformations in Speech Recognition. In *Proceedings of IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU-09)*, pages 76–81, 2009.

[TKMS03] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Vap82] V. N. Vapnik. *Estimation of Dependences based on Empirical Data*. Springer-Verlag, New York, NY, 1982.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, NY, 1998.

[VG04] K. Visweswariah and R. Gopinath. Adaptation of front end parameters in a speech recognizer. In *Proceedings of the International Conference on Spoken Language Processing*, pages 21–24, 2004.

[VOWY97] V Valtchev, J. J. Odell, P.C Woodland, and S.J Young. MMIE training of large vocabulary recognition systems. *Speech Communication*, 22(4):303–314, September 1997.

[WBS06] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

[WEK+09] Martin Wollmer, Florian Eyben, Joseph Keshet, Alex Graves, Bjorn Schuller, and Gerhard Rigoll. ROBUST DISCRIMINATIVE KEYWORD SPOTTING FOR EMOTIONALLY COLORED SPONTANEOUS SPEECH USING BIDIRECTIONAL LSTM NETWORKS. In

Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP-09), pages 3949–3952, 2009.

[WP00] P. C. Woodland and D. Povey. Large scale discriminative training for speech recognition. In *Proceedings of Automatic Speech Recognition (ASR-2000)*, pages 7–16, 2000.

[WW99] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the seventh European symposium on artificial neural networks*, volume 4, 1999.

[YDHA07] D. Yu, L. Deng, X. He, and A. Acero. Large-margin minimum classification error training for large-scale speech recognition tasks. In *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-07)*, volume 4, pages 1137–1140, 2007.

[You92] Steve J. Young. The general use of tying in phoneme-based HMM speech recognisers. In *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-92)*, pages 569–572, 1992.

[You96] Steve J. Young. Large Vocabulary Continuous Speech Recognition: A Review. *IEEE Signal Processing Magazine*, 13(5):45–68, 1996.

[YOW94] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology (HLT-94)*, pages 307–312. Association for Computational Linguistics, 1994.

[Zha04] Tong Zhang. Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms. In *Proceedings of the twenty-first international conference on Machine learning (ICML-04)*, pages 919–926, 2004.

[ZM05] Bing Zhang and Spyros Matsoukas. MINIMUM PHONEME ERROR BASED HETEROSCEDASTIC LINEAR DISCRIMINANT ANALYSIS FOR SPEECH RECOGNITION. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-05)*, number 4, pages 925–928, 2005.

[ZSG90] Victor Zuea, Stephanie Seneffa, and James Glassa. Speech database development at MIT: Timit and beyond. *Speech Communication*, 9(4):351–356, 1990.