A Framework for Speech Recognition using Logistic Regression

Øystein Birkenes

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department of Electronics and Telecommunications
Norwegian University of Science and Technology

2007

Abstract

Although discriminative approaches like the support vector machine or logistic regression have had great success in many pattern recognition application, they have only achieved limited success in speech recognition. Two of the difficulties often encountered include 1) speech signals typically have variable lengths, and 2) speech recognition is a sequence labeling problem, where each spoken utterance corresponds to a sequence of words or phones.

In this thesis, we present a framework for automatic speech recognition using logistic regression. We solve the difficulty of variable length speech signals by including a mapping in the logistic regression framework that transforms each speech signal into a fixed-dimensional vector. The mapping is defined either explicitly with a set of hidden Markov models (HMMs) for the use in penalized logistic regression (PLR), or implicitly through a sequence kernel to be used with kernel logistic regression (KLR). Unlike previous work that has used HMMs in combination with a discriminative classification approach, we jointly optimize the logistic regression parameters and the HMM parameters using a penalized likelihood criterion. Experiments show that joint optimization improves the recognition accuracy significantly. The sequence kernel we present is motivated by the dynamic time warping (DTW) distance between two feature vector sequences. Instead of considering only the optimal alignment path, we sum up the contributions from all alignment paths. Preliminary experiments with the sequence kernel show promising results.

A two-step approach is used for handling the sequence labeling problem. In the first step, a set of HMMs is used to generate an N-best list of sentence hypotheses for a spoken utterance. In the second step, these sentence hypotheses are rescored using logistic regression on the segments in the N-best list. A garbage class is introduced in the logistic regression framework in order to get reliable probability estimates for the segments in the N-best lists. We present results on both a connected digit recognition task and a continuous phone recognition task.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* (PhD) at the Norwegian University of Science and Technology (NTNU). My main supervisor has been Associate Professor Magne Hallstein Johnsen at the Department of Electronics and Telecommunications, NTNU. My co-supervisor has been Dr. Tor André Myrvoll, who was affiliated with the Department of Electronics and Telecommunications, NTNU until January 2007. He is now with SINTEF, Trondheim.

The work was carried out in the period from January 2003 to April 2007. In addition to the research activity, the work included the equivalent of one year of full-time course studies, as well as one year of teaching assistant duties. I spent most of my time with the Signal Processing Group at the Department of Electronics and Telecommunications, NTNU, but I also had the chance to do research abroad twice. In the period from March to December 2005 I visited The Institute of Statistical Mathematics (ISM), Tokyo, Japan, under the supervision of Associate Professor Tomoko Matsui. I went there again the following year, from June to August 2006.

The work was funded by a scholarship from the Research Council of Norway through the BRAGE project, which is a part of the language technology programme KUNSTI. For my first stay in Japan, I was supported by the Japan Society for the Promotion of Science (JSPS) Postdoctoral Fellowship for Foreign Researchers and JSPS Grant-in-Aid for Scientific Research (B) 16300036 and (C) 16500092. For my second stay in Japan, I received a grant from the Scandinavia-Japan Sasakawa Foundation (SJSF).

Acknowledgements

First, I would like to thank my supervisor, Associate Professor Magne Hallstein Johnsen, for his guidance and suggestions. I would also like to thank my co-supervisor, Dr. Tor André Myrvoll, for his invaluable help and for all our technical and non-technical discussions on various topics like Mac,

Japan, food, etc.

A large portion of the research leading up to this thesis was conducted while I visited the Institute of Statistical Mathematics (ISM) in Tokyo, Japan. I am indebted to Associate Professor Tomoko Matsui, for being my host at ISM in Japan, for introducing me to the exciting field of logistic regression, and for continuously motivating me. I am very grateful to Professor Kunio Tanabe for our enjoyable meetings where he taught me many things about statistics in a way that I could easily understand. I would also like to thank my office mate and friend Dr. Marco Cuturi for fruitful collaboration, and Dr. Stéphane Sénécal for our friendship and technical discussions.

In my last year as a PhD student, I was fortunate to meet Dr. Sabato Marco Siniscalchi, with whom I enjoyed enlightening discussions and collaboration. He also gave me a thorough review of my thesis. Other reviewers of my thesis that I particularly would like to thank are my fellow PhD students Svein Gunnar Pettersen, Andreas Egeberg, and Trond Skogstad.

There are many people at the Signal Processing Group at NTNU that I would like to thank. In particular, I would like to thank my office mate through more than two years, Vidar Markhus. Together we discussed many ideas on how to improve speech recognition. I am also grateful to Ole Morten Strand for our numerous discussions on speech recognition. Among other people that I would like to thank are my colleagues and friends Greg Harald Håkonsen, Fredrik Hekland, Anna Na Kim, Sébastien de la Kethulle de Ryhove, Bojana Gajić, Saeeid Tahmasbi Oskuii, Gang Lin, Duc Van Duong, Dyre Meen, and Pierluigi Salvo Rossi.

Finally, I would like to thank my parents for their moral support and for always being there for me, and my dear Yoshiko for all her love and encouragement.

Oslo, July 2007
Øystein Birkenes

Contents

Notation and Symbols

x	Scalars are typeset in non-bold lowercase
$\boldsymbol{x}$	Vectors are typeset in bold lowercase
$\boldsymbol{X}$	Vector sequences and matrices are typeset in bold uppercase (in Chapters 1 and 2, $\boldsymbol{X}$ is more general and is taken to mean an object of any type)
$\boldsymbol{W}^{\mathrm{T}}$	The transpose of $\boldsymbol{W}$
$\lvert\boldsymbol{W}\rvert$	The determinant of $\boldsymbol{W}$
$\lVert\boldsymbol{W}\rVert$	The Euclidean norm of $\boldsymbol{W}$
$\operatorname{trace}\boldsymbol{W}$	The trace of square matrix $\boldsymbol{W}$, which is the sum of the diagonal elements
$\operatorname{vec}\boldsymbol{W}$ or $\vec{\boldsymbol{W}}$	The columns of matrix $\boldsymbol{W}$ stacked into a vector
$\operatorname{diag}\boldsymbol{w}$	A diagonal matrix containing the elements of vector $\boldsymbol{w}$
$\nabla_{\boldsymbol{W}}$	The gradient matrix of partial derivatives wrt. $\boldsymbol{W}$
$\nabla^2_{\boldsymbol{W}}$	The Hessian matrix of second partial derivatives wrt. $\boldsymbol{W}$
$\otimes$	The Kronecker product
$\propto$	Proportional to
$\mathbb{R}$	The set of real numbers
$\mathbb{N}$	The set of natural numbers

Chapter 1

Introduction

Automatic speech recognition is the task of automatically converting speech into text. It has applications in many areas, including dictation, command and control, and automatic telephone services, just to name a few. The problem is far from trivial, a statement that is supported by the vast amount of research publications within the area of speech recognition over the last few decades.

The most popular approach to speech recognition is the *hidden Markov model* (HMM) framework [Rabiner, 1989]. Although the HMM framework has a range of attractive features for speech recognition, it also has some shortcomings that limit the achievable recognition performance. First, the HMM makes some incorrect assumptions about the speech signal and is therefore not the true model for speech. Second, the HMM parameters are typically estimated using the maximum likelihood (ML) criterion. This means that the parameters for each class are estimated independently of the other classes so as to best describe the generation of the observations. This is different from minimizing the probability of recognition error, which is the ultimate goal of speech recognition, and ML is therefore suboptimal. Moreover, there is no straightforward way of obtaining a confidence measure for the recognition decision.

Penalized logistic regression (PLR) [Hoerl and Kennard, 1970; Anderson and Blair, 1982] and *kernel logistic regression* (KLR) [Green and Yandell, 1985; Jaakkola and Haussler, 1999b], collectively referred to as *logistic regression*, are statistically well-founded classification approaches. Although the methods have been around for a while, they have not been particularly popular for pattern recognition applications lately, partly due to the success of the *support vector machine* (SVM) [Vapnik, 1995; Schölkopf and Smola, 2002]. Recent work [Jaakkola and Haussler, 1999b; Zhu and Hastie, 2001, 2005] has shown that logistic regression has many similar theoretical prop-

erties as SVM, and it is comparable to SVM when it comes to classification accuracy. Unlike SVM, however, logistic regression outputs the conditional probability of a class given an observation, and has a natural generalization to the multi-class case. These additional features are important in many practical pattern recognition problems, including speech recognition.

If we attempt to design a speech recognizer with logistic regression (or SVM), we face two major difficulties. First, speech signals typically have *variable lengths*, even repeating utterances of the same word from the same speaker, while logistic regression is a static classifier, meaning that the observations are assumed to be fixed-dimensional vectors. Second, a speech signal corresponds to a sequence of words, and each word corresponds to an unknown portion of the speech signal. Thus, speech recognition is a *sequence labeling problem*, with unknown segmentation, while logistic regression is designed to predict a single label only.

In this thesis we present a framework for automatic speech recognition using logistic regression. We solve the problem with variable-length speech signals by including a mapping in the logistic regression framework. The mapping maps a variable-length speech signal into a fixed-dimensional vector and is defined either explicitly with the use of a set of HMMs, or implicitly through a kernel function. A two-step approach is chosen for handling the sequence labeling problem. In the first step, a set of HMMs is used to generate an N-best list of sentence hypotheses for a spoken utterance. In the second step, these sentence hypotheses are rescored using logistic regression on the segments in the N-best list. A series of experiments demonstrate the power and show the potential of the framework.

We start this introductory chapter by giving a short review of pattern recognition and classification. Then, in Section 1.2 we list the major contributions of this thesis, and in Section 1.3 we compare the contributions with related work. Finally, in Section 1.4 we give an overview of the outline of the rest of the thesis.

1.1 Pattern Recognition and Classification

Automatic speech recognition is essentially a *pattern recognition* problem, where the goal is to recognize patterns in speech as words. In general, pattern recognition involves preprocessing, feature extraction and classification [Devroye et al., 1996; Duda et al., 2001]. In this thesis, we will rely on existing preprocessing and feature extraction methods for speech recognition, and focus our attention only on the classification step.

In statistical classification, we assume that $(\boldsymbol{X}, y) \in \mathcal{X} \times \mathcal{Y}$ is a random pair drawn according to a probability distribution $p(\boldsymbol{X}, y)$. We consider

$\boldsymbol{X} \in \mathcal{X}$ to be a suitable representation of an *observable* pattern, and $y \in \{1, \ldots, C\}$ to be an *unobservable* label describing which class the pattern belongs to. For example, $\boldsymbol{X}$ can be a sequence of Mel-frequency cepstral coefficients (MFCC) and y can represent the word label of $\boldsymbol{X}$. The goal in classification is to construct a *decision rule* h from the input space $\mathcal{X}$ to the set of class labels $\mathcal{Y}$ which is optimal in some sense. Mathematically, a decision rule can be written as

$$h : \mathcal{X} \to \mathcal{Y} \tag{1.1}$$

$$\boldsymbol{X} \mapsto \hat{y}, \tag{1.2}$$

where $\hat{y} = h(\boldsymbol{X})$ for any $\boldsymbol{X} \in \mathcal{X}$. Usually we are interested in the decision rule which gives the least probability of misclassification.

If the true distribution $p(\boldsymbol{X}, y)$ were known, the optimal decision rule would be the *Bayes decision rule* [Berger, 1985], which is

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(y|\boldsymbol{X}) \tag{1.3}$$

$$= \arg\max_{y \in \mathcal{Y}} p(\boldsymbol{X}|y)p(y). \tag{1.4}$$

In the second equality above, we have used Bayes rule and omitted the denominator $p(\boldsymbol{X})$ since it is independent of y and does not contribute to the decision.

In practical problems we do not know the true distribution $p(\boldsymbol{X}, y)$, so other decision rules are called for. In this thesis we will focus on the construction of a decision rule by the use of a finite set $\mathcal{D} = \{(\boldsymbol{X}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, y^{(N)})\}$ of samples assumed to be drawn independently according to $p(\boldsymbol{X}, y)$. These samples are called the *training data* and the approach taken to infer a decision rule from training data is known as *supervised learning*.

Within the framework of supervised learning there are two main approaches. The first approach, known as the *generative approach*, aims at modeling the joint *generative model* $p(\boldsymbol{X}, y)$ of observations $\boldsymbol{X}$ and classes y. This is usually done through modeling of the class-conditional distributions $p(\boldsymbol{X}|y)$ and the class prior $p(y)$. The estimated distributions are substituted for the true distributions in (1.4). Conventional speech recognition is done using this approach, where each observation $\boldsymbol{X}$ is a sequence of feature vectors extracted from a speech signal. The class-conditional distributions $p(\boldsymbol{X}|y)$ are then typically obtained from HMMs.

The other main approach to supervised learning is called the *discriminative approach*. In this approach, the conditional distribution $p(y|\boldsymbol{X})$ of class labels given an observation is modeled and substituted for the true one

in (1.3). Logistic regression is an example of this approach. Other discriminative approaches include kernel methods such as SVM, which approximate the decision rule in (1.3) in a discriminative way without explicitly providing the conditional probabilities $p(y|\boldsymbol{X})$.

Both the generative approach and the discriminative approach to learning classifiers have their strengths and weaknesses [Ng and Jordan, 2002; Ulusoy and Bishop, 2005]. The generative approach can handle missing or partially labeled data, and can make use of a combination of small amounts of expensive labeled training data with large quantities of cheap unlabeled training data. Moreover, generative models can readily handle compositionality, which in speech recognition terms means that long linguistic units such as words can be modeled by concatenating a set of models for short linguistic units such as phones. On the other hand, in the discriminative approach the decision rule is learnt in a more direct way than in the generative approach, without making assumptions on the distribution of the observations. Modeling effort is spent in the confusable regions of the observation space that are important for classification, whereas in the generative approach an attempt is made to accurately model regions in the observation space that may be irrelevant for the outcome of the resulting decision rule.

Since the generative approach and the discriminative approach have complementary strengths and weaknesses, it seems natural to attempt to combine them. *Discriminative training* is one such approach, where a generative model is trained using a discriminative criterion function. Examples of discriminative training approaches that have been successfully applied to speech recognition include *maximum mutual information* (MMI) [Bahl et al., 1986] and *minimum classification error* (MCE) [Juang et al., 1997]. Another combination approach is to use a discriminative classifier such as logistic regression or SVM that incorporates a generative model [Jaakkola and Haussler, 1999a; Smith and Gales, 2002]. The latter is the approach taken in this thesis.

1.2 Contributions of this Thesis

This thesis provides a study of logistic regression and its use in automatic speech recognition. Both penalized logistic regression (PLR) and kernel logistic regression (KLR) are considered. Contributions are made in the logistic regression framework, as well as in the application to isolated-word speech recognition and N-best rescoring for continuous speech recognition. Parts of the work has been published in [Birkenes et al., 2005, 2006a,b, 2007; Cuturi et al., 2007].

1.2.1 Contributions to the Logistic Regression Framework

While most presentations of logistic regression in the literature assume the observations to be fixed-dimensional vectors (e.g., [Jaakkola and Haussler, 1999b; Tanabe, 2001a,b; Zhu and Hastie, 2001, 2005]), we present a more general logistic regression framework that allows observations to be objects of arbitrary type. The generalization is trivial, however, since we simply redefine a nonlinear mapping such that it maps an observation of any particular type into a fixed-dimensional vector. The motivation for this simple generalization is that we would like to use logistic regression for speech recognition, where each observation is a sequence of feature vectors or a *time series*.

In this thesis, we also provide necessary proofs of derivatives in the logistic regression framework, many of which are missing in the literature. In addition, we provide derivations of the formulas for the determination of the regularization parameter in PLR with the use of a Bayesian information criterion (ABIC) [Akaike, 1980].

The two major contributions to the logistic regression framework, however, are 1) *adaptive regressor parameters*, which allows for more flexible decision boundaries for PLR, and 2) a *garbage class*, to ensure proper behaviour for the prediction of atypical observations for both PLR and KLR.

1.2.2 Isolated-Word Speech Recognition using Logistic Regression

The major difficulty in applying logistic regression to isolated-word speech recognition is that speech signals typically have variable lengths. We present solutions to this problem in the form of several mappings from variable-length time series into fixed-dimensional vectors. The mappings are either defined explicitly, as in PLR, or implicitly through a kernel to be used with KLR. The mappings for PLR are based on a set of HMMs. They include a mapping of a time series into a vector of the likelihoods of each HMM, and a mapping into a vector of likelihood-ratios of each model and its corresponding anti-model. With the use of adaptive regressor parameters, which in this case are the HMM parameters, we obtain a powerful discriminative classifier for speech signals that combines the advantages of the generative learning approach and the discriminative learning approach.

We present two families of kernels to be used with KLR. The first family consists of vector kernels (e.g., Gaussian or polynomial) that operate on pairs of time series via the fixed-dimensional vectors obtained from either of the explicit mappings. The second family are sequence kernels that operate directly on pairs of time series. A particular member of this family

is the *global alignment (GA) kernel* [Cuturi et al., 2007], which we recently proposed. With the latter family of kernels, KLR is a purely discriminative approach for isolated-word speech recognition.

Experiments on the E-set of the TI46 database compare the various approaches. Also, a phone classification experiment on the TIMIT database [Lamel et al., 1986] using PLR with adaptive regressor parameters is done.

1.2.3 N-Best Rescoring using Logistic Regression on Segments

We choose a two-step approach to continuous speech recognition. In the first step, a set of HMMs is used to generate an N-best list of sentence hypotheses for a spoken utterance. In the second step, these sentence hypotheses are rescored using logistic regression on the segments in the N-best list. The new sentence scores are either used directly to reorder the sentence hypotheses in the N-best list, or they are interpolated with the HMM likelihoods of the corresponding sentence hypotheses before reordering. We argue that logistic regression with a garbage class is necessary in this approach.

The N-best rescoring approach is tested on the Aurora2 database [Pearce and Hirsch, 2000] for connected digit recognition. We also perform continuous phone recognition using the TIMIT database [Lamel et al., 1986].

1.3 Related Work

In this section, we mention various papers that are related to the work presented in this thesis. We start with papers that use a set of HMMs as a preprocessor for logistic regression or SVM in order to perform isolated-word speech recognition. Then we cite two papers that use sequence kernels that operate directly on pairs of time series. Finally, we mention a paper that resembles the way we do continuous speech recognition.

Perhaps the first paper to use HMMs in order to map time series into fixed-dimensional vectors for the use in logistic regression and SVM was [Jaakkola and Haussler, 1999a]. Their presentation was general, however, in the sense that they targeted a range of pattern recognition applications, and not only speech recognition with HMMs. They proposed the *Fisher score mapping*, which maps a time series into the gradient space of a single HMM. Furthermore, they constructed a kernel from the Fisher score mapping, known as the *Fisher kernel*. The kernel was used in binary SVM and two-class KLR. In their approach, the HMM and the discriminative classifier

were trained separately with the use of two different criteria. This is also the case for the other methods presented in this section.

In [Smith and Gales, 2002], the authors considered the use of the Fisher score in speech recognition. They used binary SVM and considered the use of two HMMs instead of only one HMM as in [Jaakkola and Haussler, 1999a]. Moreover, they proposed to append the Fisher score with the likelihood, likelihood-ratio, or even higher order derivatives of the HMM parameters.

In [Layton and Gales, 2006] the authors built on the work in [Smith and Gales, 2002] and introduced the *conditional augmented (C-Aug) model*. The C-Aug model resembles to a high degree the multinomial logistic regression model presented in this thesis. A set of one HMM for each class is used in the mapping, but unlike our approach, where the model for each class probability depends on all HMMs, the C-Aug model for a class probability depends only on the HMM for that particular class. Moreover, training of the C-Aug model is done using the maximum likelihood criterion without a penalty term.

In [Abou-Moustafa et al., 2004], the authors presented a class-independent mapping that makes use of the HMMs of all the classes. In their approach, each element of the mapped observation is the log-likelihood of a HMM. The authors used SVM as the discriminative classifier, and presented results on a handwriting recognition task.

For the purely discriminative approach of using a sequence kernel for speech recognition, there is not much to find in the literature. Notable papers include [Shimodaira et al., 2002] and [Bahlmann et al., 2002]. The former paper introduced the *dynamic time alignment kernel* (DTAK) for applications in speech recognition, while the latter paper introduced a kernel for the use in handwriting recognition. In both papers, they introduced a sequence kernel directly operating on pairs of time series. The kernels are motivated by the *dynamic time warping* (DTW) [Rabiner and Juang, 1993] algorithm. Both kernels are defined as the alignment score along the optimal alignment path, and they are not positive definite in general. In this thesis we present the *global alignment (GA) kernel* [Cuturi et al., 2007]. The GA kernel sums up the contributions for all the alignment paths, and it can be shown to be positive definite under mild conditions.

In [Ganapathiraju et al., 2004], the authors presented a hybrid HMM/SVM approach for continuous speech recognition using the N-best rescoring paradigm. Their method addressed both the issue of variable-length sequences and the issue of sequence labeling with unknown segmentation, but it had several weaknesses. Since the problem of segments (phones) with varying lengths was solved by discarding all but a fixed number of feature vectors, much information in the speech signals was lost.

Moreover, in the rescoring of the N-best lists, sentences with deletion and insertion errors could not be corrected. In this thesis we introduce the concept of a *garbage class* in order to rescore all hypotheses in an N-best lists, which implies that substitution errors, insertion errors and deletion errors may be corrected.

1.4 Outline

The outline of the thesis is as follows. In Chapter 2 we present logistic regression, including PLR and KLR. Chapter 3 is a review of HMMs and the conventional way of doing automatic speech recognition. In Chapter 4 we consider the application of logistic regression to isolated-word speech recognition. Chapter 5 is devoted to the application of logistic regression to N-best rescoring. Finally, Chapter 6 contains the conclusions and a section about future work.

Chapter 2

Logistic Regression

We use the term *logistic regression* to refer to both *penalized logistic regression* (PLR) and its dual formulation which is called *dual penalized logistic regression* (dPLR), or more commonly *kernel logistic regression* (KLR). Many authors have presented the framework of logistic regression in the context of multiclass classification [Jaakkola and Haussler, 1999b; Tanabe, 2001a,b; Zhu and Hastie, 2001, 2005]. The various authors use different approaches to explain the theory. In this chapter we present the theory of both PLR and KLR using mostly the approach taken in [Tanabe, 2001a,b]. Our presentation is somewhat more general, however, in that we allow the inputs to be of arbitrary form, and not only fixed-dimensional vectors which is often assumed. We do this in order to prepare for the following chapters, where the inputs are sequences of feature vectors extracted from speech signals.

We start by explaining the PLR in Section 2.1. In Section 2.1.1 we consider a method for determining the regularization parameter, and in sections 2.1.2 and 2.1.3 we introduce the concepts of a *garbage class* and *adaptive regressor parameters*, respectively. We present KLR in Section 2.2. Finally, Section 2.3 contains a short summary.

2.1 Penalized Logistic Regression

In this section we will see how penalized logistic regression (PLR) can be used to estimate the conditional probability distribution $p(y|\boldsymbol{X})$ of a class label $y \in \mathcal{Y}$ given an observation $\boldsymbol{X} \in \mathcal{X}$. Classification is accomplished by selecting the class label $\hat{y}$ giving the largest conditional probability, that is,

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(y|\boldsymbol{X}). \tag{2.1}$$

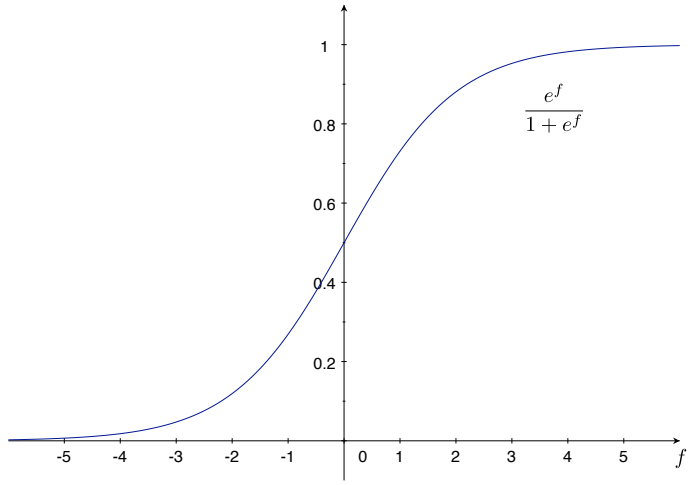

Figure 2.1: The logistic function.

In the following, we introduce a parametric model for $p(y|\boldsymbol{X})$. Then we define the criterion function which we will use in order to estimate the parameters of the model, followed by an optimization algorithm specifically designed for this purpose.

Before presenting the general form of the logistic regression model that we will use in this thesis as a model for $p(y|\boldsymbol{X})$, let us start by considering the simple case of $C = 2$ classes and real D-dimensional feature vectors $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^D$. A popular model for the conditional probability of class $y = 1$ given $\boldsymbol{x}$ is

$$p(y = 1|\boldsymbol{x}) = \frac{e^f}{1 + e^f}, \tag{2.2}$$

where the *discriminant function* $f = w_1 + w_2 x_1 + \cdots + w_{D+1} x_D$ is a linear combination (plus a bias term) of the elements of $\boldsymbol{x}$ with parameters $w_1, \ldots, w_{D+1}$. Usually, the discriminant function is written as the inner product $f = \boldsymbol{w}^{\mathrm{T}} \bar{\boldsymbol{x}}$, where $\bar{\boldsymbol{x}}$ is the vector $\boldsymbol{x}$ augmented with a "1" before the first element, and $\boldsymbol{w}$ is a *weight vector* that serves as the parameter vector of the model. Since the conditional probability of the two classes must sum to one, the conditional probability for the other class is $p(y = 2|\boldsymbol{x}) = 1 - p(y = 1|\boldsymbol{x})$. The function in (2.2) is known as the *logistic function*, or *binomial logistic regressor*, and, apart from being just a squashing function that maps f into the interval $[0, 1]$, it also has good probabilistic properties in the context of classification [Jordan, 1995]. The graphic representation of the logistic function is shown in Figure 2.1.

The natural extension to classification problems with more than two classes is to model the conditional probabilities with the *softmax function*

or *multinomial logistic regressor* defined by

$$p(y = i|\boldsymbol{x}) = \frac{e^{f_i}}{\sum_{j=1}^{C} e^{f_j}} \quad \text{for } i = 1, \ldots, C, \tag{2.3}$$

where $f_i = \boldsymbol{w}_i^{\mathrm{T}} \bar{\boldsymbol{x}}$ is the discriminant function for class i parameterized by the weight vector $\boldsymbol{w}_i$. Due to the probability constraint $\sum_{i=1}^{C} p(y = i|\boldsymbol{x}) = 1$, the weight vector for one of the classes, say $\boldsymbol{w}_C$, need not be estimated and can be set to all zeros. In this thesis however, we follow the convention in [Tanabe, 2001a] and keep the redundant representation with C non-zero weight vectors. As explained in [Tanabe, 2001a], this is done for numerical stability reasons, and in order to treat all the classes equally. We let each weight vector be a column of the matrix

$$\boldsymbol{W} = \begin{bmatrix} | & & | \\ \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_C \\ | & & | \end{bmatrix}, \tag{2.4}$$

which is the *parameter matrix* of the model in (2.3).

With discriminant functions that are linear in the feature vectors, only linear decision boundaries between the classes in the feature space $\mathcal{X} = \mathbb{R}^D$ can be found. A simple way to achieve nonlinear decision boundaries is to first map the features into an $M + 1$-dimensional Euclidean space using a nonlinear mapping

$$\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^{M+1} \tag{2.5}$$

$$\boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x}), \tag{2.6}$$

where we let the first element of the mapping be a "1" in order to accommodate a bias term. Then, we define the nonlinear discriminant functions to be used in (2.3) as

$$f_i = \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{\lambda}) \quad \text{for } i = 1, \ldots, C, \tag{2.7}$$

where $\boldsymbol{\lambda}$ is a hyperparameter vector of the mapping $\boldsymbol{\phi}$. Each element of the vector $\boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{\lambda})$ is a function of $\boldsymbol{x}$. These functions are called *regressors* and they play an important role in the logistic regression framework.

A straightforward generalization to the logistic regression framework can be obtained by redefining the mapping in (2.5) to

$$\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^{M+1} \tag{2.8}$$

$$\boldsymbol{X} \mapsto \boldsymbol{\phi}(\boldsymbol{X}), \tag{2.9}$$

where $\mathcal{X}$ is the set of observations of arbitrary type. We have written $\boldsymbol{X}$ in the above equation in place of $\boldsymbol{x}$ in order to emphasize that we are no longer

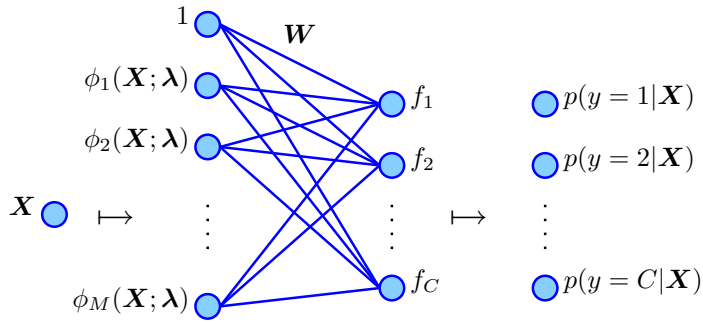

Figure 2.2: The logistic regression model.

restricted to real-valued feature vectors. In fact, as long as a mapping $\boldsymbol{\phi}$ can be found, $\mathcal{X}$ can be any nonempty set.

To summarize, we have introduced the following model for the conditional probability of class $y = i$ given $\boldsymbol{X}$:

$$p(y = i|\boldsymbol{X}, \boldsymbol{W}) = \frac{e^{\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}}{\sum_{j=1}^{C} e^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}} \quad \text{for } i = 1, \ldots, C, \tag{2.10}$$

where $\boldsymbol{W}$ is an $(M + 1) \times C$ parameter matrix with columns $\boldsymbol{w}_i$, and $\boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})$ is a vector of $M + 1$ regressors with hyperparameter $\boldsymbol{\lambda}$, with the first regressor being the constant "1". We will refer to this model as the *multinomial logistic regression model*, or simply as the *logistic regression model*. Figure 2.2 illustrates the model.

The classical way to estimate the parameter matrix $\boldsymbol{W}$ from a set of training data $\mathcal{D} = \{(\boldsymbol{X}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, y^{(N)})\}$ is to maximize the *likelihood*

$$L(\boldsymbol{W}; \mathcal{D}) = \prod_{n=1}^{N} p(y = y^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{W}). \tag{2.11}$$

However, the maximum likelihood estimate does not always exist [Albert, A. and Anderson, J. A., 1984]. This happens, for example, when the mapped data set $\{(\boldsymbol{\phi}(\boldsymbol{X}^{(1)};\boldsymbol{\lambda}), y^{(1)}), \ldots, (\boldsymbol{\phi}(\boldsymbol{X}^{(N)};\boldsymbol{\lambda}), y^{(N)})\}$ is linearly separable. Moreover, even though the maximum likelihood estimate exists, overfitting to the training data may occur, which in turn leads to poor generalization performance. For that reason, we introduce a *penalty* $\pi(\boldsymbol{W})$ on the parameters and find an estimate $\hat{\boldsymbol{W}}$ by maximizing the *penalized likelihood*

$$\mathcal{P}_\delta(\boldsymbol{W}; \mathcal{D}) = L(\boldsymbol{W}; \mathcal{D})\pi^\delta(\boldsymbol{W}), \tag{2.12}$$

where $\delta \geq 0$ is a hyperparameter used to balance the likelihood and the penalty factor. There are many ways to define the penalty factor. In this

thesis we follow [Tanabe, 2001a] and use a penalty of the form

$$\pi(\boldsymbol{W}) = \prod_{i=1}^{C} e^{-\frac{\gamma_i}{2}\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{w}_i} \tag{2.13}$$

$$= e^{-\frac{1}{2}\sum_{i=1}^{C}\gamma_i\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{w}_i} \tag{2.14}$$

$$= e^{-\frac{1}{2}\operatorname{trace}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{W}}, \tag{2.15}$$

where $\boldsymbol{\Sigma}$ is an $(M+1)\times(M+1)$ positive definite matrix, and $\boldsymbol{\Gamma}$ is a $C \times C$ diagonal matrix with elements $\gamma_1, \ldots, \gamma_C$, that is,

$$\boldsymbol{\Gamma} = \begin{bmatrix} \gamma_1 & & 0 \\ & \ddots & \\ 0 & & \gamma_C \end{bmatrix}. \tag{2.16}$$

In [Tanabe, 2001a], the author discusses various choices for the matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma}$. One such choice that we will adopt in this thesis is explained in the following. The $\boldsymbol{\Gamma}$ matrix should compensate for differences in the number of training examples from each class, as well as include prior probabilities for the various classes. If we let N_i denote the number of training examples from class i, and $p(y=i)$ denote our belief in the prior probability for class i, we let the ith element of $\boldsymbol{\Gamma}$ be

$$\gamma_i = \frac{N_i}{Np(y=i)}. \tag{2.17}$$

We let $\boldsymbol{\Sigma}$ be the sample moment matrix of the transformed observations $\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})$ for $n = 1, \ldots, N$, that is,

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda}) \tag{2.18}$$

$$= \frac{1}{N}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}, \tag{2.19}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} | & & | \\ \boldsymbol{\phi}^{(1)} & \cdots & \boldsymbol{\phi}^{(N)} \\ | & & | \end{bmatrix} \tag{2.20}$$

is the $(M+1)\times N$-matrix whose nth column is $\boldsymbol{\phi}^{(n)} = \boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})$.

It is insightful to note that the above penalized likelihood parameter estimation procedure can also be interpreted in a Bayesian way, as maximum a posteriori (MAP) estimation. This is the preferred method for

some authors (e.g., [Krishnapuram et al., 2005]). In the Bayesian formalism, the parameter matrix $\boldsymbol{W}$ is considered to be a random quantity, with a prior distribution $p(\boldsymbol{W})$. If we choose $p(\boldsymbol{W}) \propto \pi^{\delta}(\boldsymbol{W})$, we can see from (2.13) that our choice of prior over the matrix parameter $\boldsymbol{W}$ is the product of priors over the vectors $\boldsymbol{w}_i$, where each vector $\boldsymbol{w}_i$ has a multivariate normal distribution with zero mean and precision matrix $\delta\gamma_i\boldsymbol{\Sigma}$. The variance-covariance matrix is the inverse of the precision matrix and is thus $1/(\delta\gamma_i)\boldsymbol{\Sigma}^{-1}$. In MAP estimation, the prior $p(\boldsymbol{W})$ is multiplied with the likelihood $L(\boldsymbol{W};\mathcal{D})$ to form a quantity which is proportional to the posterior. This is the quantity which is to be maximized, and since it is the same as the penalized likelihood in (2.12), MAP estimation and maximum penalized likelihood estimation give the same result.

Maximizing the penalized likelihood in (2.12) is mathematically equivalent to minimizing the negative logarithm of the penalized likelihood, which can be written

$$\begin{aligned}
\mathcal{P}_{\delta}^{\log}(\boldsymbol{W};\mathcal{D}) &= -\log \mathcal{P}_{\delta}(\boldsymbol{W};\mathcal{D}) && (2.21)\\
&= -\log L(\boldsymbol{W};\mathcal{D}) - \delta \log \pi(\boldsymbol{W}) && (2.22)\\
&= -\sum_{n=1}^{N} \log p(y = y^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{W}) + \frac{\delta}{2} \operatorname{trace} \boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{W}. && (2.23)
\end{aligned}$$

In the following, we will be concerned with the minimization of the above criterion function. Let us start with two lemmas that give expressions for the gradient and the Hessian of (2.23). These expressions will be used to design an optimization algorithm in order to find an estimate of $\boldsymbol{W}$.

Lemma 2.1.1 *The gradient of* (2.23) *is the* $(M+1) \times C$ *matrix*

$$\nabla_{\boldsymbol{W}} \mathcal{P}_{\delta}^{\log}(\boldsymbol{W};\mathcal{D}) = \boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{W}\boldsymbol{\Gamma}, \tag{2.24}$$

where

$$\boldsymbol{P}(\boldsymbol{W}) = \begin{bmatrix} | & & | \\ \boldsymbol{p}^{(1)} & \cdots & \boldsymbol{p}^{(N)} \\ | & & | \end{bmatrix} \tag{2.25}$$

is a $C \times N$ *matrix whose* n*th column is a vector of the conditional probabilities for all classes given* $\boldsymbol{X}^{(n)}$*, i.e.,* $\boldsymbol{p}^{(n)} = [p(y^{(n)} = 1|\boldsymbol{X}^{(n)}, \boldsymbol{W}), \ldots, p(y^{(n)} = C|\boldsymbol{X}^{(n)}, \boldsymbol{W})]^{\mathrm{T}}$*, and*

$$\boldsymbol{Y} = \begin{bmatrix} | & & | \\ \boldsymbol{e}_{y^{(1)}} & \cdots & \boldsymbol{e}_{y^{(N)}} \\ | & & | \end{bmatrix} \tag{2.26}$$

is a $C \times N$ *matrix where the* n*th column* $\boldsymbol{e}_{y^{(n)}}$ *is a unit vector with all zeros except for element* $y^{(n)}$ *which is* 1*.*

Proof See App. A

Lemma 2.1.2 *The Hessian of* (2.23) *is*

$$\nabla^2_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}; \mathcal{D}) = \sum_{n=1}^{N} (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)} \boldsymbol{\phi}^{(n)\mathrm{T}} + \delta \boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}, \quad (2.27)$$

where $\otimes$ *is the Kronecker product.*

Proof See App. A

It can be shown [Tanabe, 2001a] that the Hessian $\nabla^2_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}; \mathcal{D})$ is a positive definite matrix. This means that $\mathcal{P}^{\log}_{\delta}(\boldsymbol{W}, \mathcal{D})$ is a convex function whose unique minimizer $\boldsymbol{W}^*$ satisfies

$$\boldsymbol{W}^* = \frac{1}{\delta} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi} (\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{P} (\boldsymbol{W}^*)^{\mathrm{T}}) \boldsymbol{\Gamma}^{-1}. \quad (2.28)$$

The above equation can be found by setting the gradient in (2.24) to zero. Note that the minimizer $\boldsymbol{W}^*$ appears on both sides of the equation, with the one on the right appearing within the nonlinear term $\boldsymbol{P}(\boldsymbol{W}^*)^{\mathrm{T}}$. Thus, the minimizer $\boldsymbol{W}^*$ cannot be found analytically, and we must rely on numerical methods to obtain an estimate.

We will here present an optimization algorithm for estimating the weight matrix $\boldsymbol{W}$ that makes use of both the gradient in (2.24) and the Hessian in (2.27). The algorithm was introduced in [Tanabe, 2001a] where it was called the *penalized logistic regression machine* (PLRM). In this algorithm, the weight matrix is updated iteratively using Newton's method, where each step is

$$\boldsymbol{W}^{i+1} = \boldsymbol{W}^i - \alpha_i \Delta \boldsymbol{W}^i, \quad (2.29)$$

where $\Delta \boldsymbol{W}^i$ is defined in

$$\operatorname{vec} \Delta \boldsymbol{W}^i = [\nabla^2_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}^i; \mathcal{D})]^{-1} \operatorname{vec} \nabla_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}^i; \mathcal{D}). \quad (2.30)$$

The factor α_i is a stepsize. In order to find the update matrix $\Delta \boldsymbol{W}^i$, the inverse of the Hessian needs to be computed, and this has to be done at every step i. Computing the inverse of the Hessian is very costly, so in [Tanabe, 2001a] the author suggested to compute an approximation to $\Delta \boldsymbol{W}^i$ using the conjugate gradient (CG) method [Hestenes and Stiefel, 1952; Luenberger, 1989]. This amounts to solving for $\Delta \boldsymbol{W}^i$ in the equation

$$\nabla^2_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}^i; \mathcal{D}) \operatorname{vec} \Delta \boldsymbol{W}^i = \operatorname{vec} \nabla_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}^i; \mathcal{D}), \quad (2.31)$$

which after substitution of (2.24) and (2.27) reduces to

$$\sum_{n=1}^{N}(\boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}})\Delta\boldsymbol{W}^i(\operatorname{diag}\boldsymbol{p}^{(n)}-\boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}})+\delta\boldsymbol{\Sigma}\Delta\boldsymbol{W}^i\boldsymbol{\Gamma}$$
$$=\boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W}^{i\mathrm{T}})-\boldsymbol{Y}^{\mathrm{T}})+\delta\boldsymbol{\Sigma}\boldsymbol{W}^i\boldsymbol{\Gamma}. \quad (2.32)$$

The CG method for solving this equation is summarized in the following algorithm, which computes an estimate of the weight matrix that minimizes the criterion function in (2.23).

Algorithm 2.1.3 (The penalized logistic regression machine [Tanabe, 2003]) *Start with an initial weight matrix* $\boldsymbol{W}^0$ *and generate a sequence of matrices according to*

$$\boldsymbol{W}^{i+1}=\boldsymbol{W}^i-\alpha_i\Delta\boldsymbol{W}^i, \quad (2.33)$$

where $\Delta\boldsymbol{W}^i$ *is computed using the following conjugate gradient method:*

1. *Initialize: Start with an initial matrix* $\Delta\boldsymbol{W}_0^i$ *and compute the matrices* $\boldsymbol{R}_0$ *and* $\boldsymbol{Q}_0$*:*

$$\boldsymbol{R}_0=\boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W}^{i\mathrm{T}})-\boldsymbol{Y}^{\mathrm{T}})+\delta\boldsymbol{\Sigma}\boldsymbol{W}^i\boldsymbol{\Gamma}$$
$$-\sum_{n=1}^{N}\boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\Delta\boldsymbol{W}_0^i(\operatorname{diag}\boldsymbol{p}^{(n)}-\boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}})-\delta\boldsymbol{\Sigma}\Delta\boldsymbol{W}_0^i\boldsymbol{\Gamma}, \quad (2.34)$$

$$\boldsymbol{Q}_0=\sum_{n=1}^{N}\boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{R}_0(\operatorname{diag}\boldsymbol{p}^{(n)}-\boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}})+\delta\boldsymbol{\Sigma}\boldsymbol{R}_0\boldsymbol{\Gamma}. \quad (2.35)$$

2. *Iterate: Generate a sequence* $(\Delta\boldsymbol{W}_1^i,\Delta\boldsymbol{W}_2^i,\dots)$ *according to*

$$\alpha_j=\frac{\|\sum_{n=1}^{N}\boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{R}_j(\operatorname{diag}\boldsymbol{p}^{(n)}-\boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}})+\delta\boldsymbol{\Sigma}\boldsymbol{R}_j\boldsymbol{\Gamma}\|^2}{\|\sum_{n=1}^{N}\boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{Q}_j(\operatorname{diag}\boldsymbol{p}^{(n)}-\boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}})+\delta\boldsymbol{\Sigma}\boldsymbol{Q}_j\boldsymbol{\Gamma}\|^2}, \quad (2.36)$$

$$\Delta\boldsymbol{W}_{j+1}^i=\Delta\boldsymbol{W}_j^i+\alpha_j\boldsymbol{Q}_j, \quad (2.37)$$

$$\boldsymbol{R}_{j+1} = \boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W}^{i\mathrm{T}}) - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{W}^{i}\boldsymbol{\Gamma} - \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\Delta\boldsymbol{W}_{j+1}^{i}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) - \delta\boldsymbol{\Sigma}\Delta\boldsymbol{W}_{j+1}^{i}\boldsymbol{\Gamma}, \tag{2.38}$$

$$\beta_{j+1} = \frac{\| \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{R}_{j+1}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{R}_{j+1}\boldsymbol{\Gamma}\|^2}{\| \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{R}_{j}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{R}_{j}\boldsymbol{\Gamma}\|^2}, \tag{2.39}$$

$$\boldsymbol{Q}_{j+1} = \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{R}_{j+1}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{R}_{j+1}\boldsymbol{\Gamma} + \beta_{j+1}\boldsymbol{Q}_{j}. \tag{2.40}$$

2.1.1 Determining the Regularization Parameter δ

An important issue in the penalized logistic regression framework is to determine the optimal value of the regularization parameter δ. This will prevent overfitting to the training data, thereby ensuring good generalization performance of the logistic regression model. In the following, we present two methods of how to estimate the optimal value of δ. The first method is to minimize the average cross-validation error, and the second method is to minimize a Bayesian information criterion (ABIC).

Minimization of the average cross-validation error

A relatively straightforward way to obtain an estimate of δ is through K-*fold cross-validation* [Duda et al., 2001]. In this method, the training set is first partitioned into K subsets, e.g., $K = 10$. Then, training of the logistic regression model is performed on the first $K - 1$ parts for various values of δ. The remaining part is used for testing or *validation*. The procedure is repeated K times, such that every subset is used once for validation. In the end, the K error rates obtained for each δ are averaged, and the δ giving the lowest average error rate is chosen as the estimate.

Minimization of a Bayesian information criterion (ABIC)

An arguably more elegant way to estimate the regularization parameter δ is to minimize *a Bayesian information criterion* (ABIC) [Akaike, 1980; Tanabe, 2001a]. The idea is to find the value of δ that maximizes the probability of the training data, without assuming any particular parameter matrix $\boldsymbol{W}$. The ABIC criterion is

$$\mathrm{ABIC}(\delta) = -2 \log p(\mathcal{D}|\delta), \tag{2.41}$$

where $p(\mathcal{D}|\delta)$ is the *marginal likelihood* defined by

$$p(\mathcal{D}|\delta) = \int p(\mathcal{D}|\boldsymbol{W}, \delta) p(\boldsymbol{W}|\delta) \mathrm{d}\boldsymbol{W}. \tag{2.42}$$

Thus, we can see that minimizing the ABIC criterion is equivalent to maximizing the marginal likelihood, which is the probability of the training data given δ.

The integrand in (2.42) is the posterior distribution of the model with likelihood $p(\mathcal{D}|\boldsymbol{W}, \delta) = L(\boldsymbol{W}; \mathcal{D})$ and prior $p(\boldsymbol{W}|\delta)$. The prior distribution can be written

$$p(\boldsymbol{W}|\delta) \propto \delta^{C(M+1)/2} \pi^{\delta}(\boldsymbol{W}), \tag{2.43}$$

where $\pi(\boldsymbol{W})$ is the penalty factor introduced in (2.12). The normalization constants that are independent of $\boldsymbol{W}$ and δ are omitted from the right hand side above since they are irrelevant in the minimization of $\mathrm{ABIC}(\delta)$. We now have

$$p(\mathcal{D}|\delta) \propto \delta^{C(M+1)/2} \int L(\boldsymbol{W}; \mathcal{D}) \pi^{\delta}(\boldsymbol{W}) \mathrm{d}\boldsymbol{W} \tag{2.44}$$

$$= \delta^{C(M+1)/2} \int \mathcal{P}_{\delta}(\boldsymbol{W}) \mathrm{d}\boldsymbol{W} \tag{2.45}$$

$$= \delta^{C(M+1)/2} \int e^{-\mathcal{P}_{\delta}^{\log}(\boldsymbol{W})} \mathrm{d}\boldsymbol{W}, \tag{2.46}$$

where $\mathcal{P}_{\delta}(\boldsymbol{W}) = \mathcal{P}_{\delta}(\boldsymbol{W}; \mathcal{D})$ is the penalized likelihood in (2.12) and $\mathcal{P}_{\delta}^{\log}(\boldsymbol{W}) = -\log \mathcal{P}_{\delta}(\boldsymbol{W})$ is the negative logarithm of the penalized likelihood. The integral in (2.46) does not have an analytic solution [Tanabe, 2001a]. An approximation to the integral is considered next.

In the following, we use the notation $\vec{\boldsymbol{W}} = \mathrm{vec}\, \boldsymbol{W}$ to denote the vectorized version of the matrix $\boldsymbol{W}$. Furthermore, let $\widetilde{\mathcal{P}}_{\delta}^{\log}(\boldsymbol{W})$ be the quadratic approximation of $\mathcal{P}_{\delta}^{\log}(\boldsymbol{W})$ about its maximizing argument $\boldsymbol{W}^*$, i.e.,

$$\widetilde{\mathcal{P}}_{\delta}^{\log}(\boldsymbol{W}) = \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}^*) + \frac{1}{2}(\vec{\boldsymbol{W}} - \vec{\boldsymbol{W}}^*)^T \nabla_{\boldsymbol{W}}^2 \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}^*)(\vec{\boldsymbol{W}} - \vec{\boldsymbol{W}}^*) \tag{2.47}$$

$$= \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}^*) + q(\boldsymbol{W}), \tag{2.48}$$

where $q(\boldsymbol{W})$ is the quadratic form defined in (2.47). The above approximation is good if $\mathcal{P}_\delta^{\log}(\boldsymbol{W})$ has a shape which is close to a quadratic form, in which case $\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W}) \approx \mathcal{P}_\delta^{\log}(\boldsymbol{W})$. With this in mind, we write the integral in (2.46) as

$$\int e^{-\mathcal{P}_\delta^{\log}(\boldsymbol{W})}\mathrm{d}\boldsymbol{W} = \int e^{-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})} e^{-\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})}\mathrm{d}\boldsymbol{W} \tag{2.49}$$

$$= \int e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} e^{-\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})}\mathrm{d}\boldsymbol{W} \tag{2.50}$$

$$= \int e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} e^{-\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)-q(\boldsymbol{W})}\mathrm{d}\boldsymbol{W} \tag{2.51}$$

$$= e^{-\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)} \int e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} e^{-q(\boldsymbol{W})}\mathrm{d}\boldsymbol{W} \tag{2.52}$$

$$= e^{-\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)} Z \int e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} f(\boldsymbol{W})\mathrm{d}\boldsymbol{W}, \tag{2.53}$$

where

$$f(\boldsymbol{W}) = \frac{1}{Z} e^{-q(\boldsymbol{W})} \tag{2.54}$$

is a Gaussian distribution with mean $\boldsymbol{W}^*$, covariance matrix $\nabla^2_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)^{-1}$, and normalization constant

$$Z = (2\pi)^{C(M+1)/2}|\nabla^2_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)|^{-1/2}, \tag{2.55}$$

with $|\cdot|$ denoting the determinant of a matrix. Now, the marginal likelihood can be written

$$p(\mathcal{D}|\delta) \propto \delta^{C(M+1)/2}|\nabla^2_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)|^{-1/2} e^{-\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)} \cdot \int e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} f(\boldsymbol{W})\mathrm{d}\boldsymbol{W}. \tag{2.56}$$

Furthermore, we can write the ABIC criterion as

$$ABIC(\delta) = -2\log p(\mathcal{D}|\delta) \tag{2.57}$$

$$\propto 2\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*) + \log|\nabla^2_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)| - C(M+1)\log\delta + \text{correction}, \tag{2.58}$$

where the correction term is

$$\text{correction} = -2\log\int e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})-\mathcal{P}_\delta^{\log}(\boldsymbol{W})} f(\boldsymbol{W})\mathrm{d}\boldsymbol{W}. \tag{2.59}$$

If $\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W})$ is a good approximation to $\mathcal{P}_\delta^{\log}(\boldsymbol{W})$, the correction term is close to zero, and an approximate criterion can be taken to be right hand side of (2.57) without the correction term.

We can approximate the correction term by generating a set of independent samples $\{\boldsymbol{W}_j\}_{j=1}^J$ from $f(\boldsymbol{W})$ which we substitute into the following expression:

$$\text{correction} \approx -2 \log \frac{1}{J} \sum_{j=1}^{J} e^{\widetilde{\mathcal{P}}_\delta^{\log}(\boldsymbol{W}_j) - \mathcal{P}_\delta^{\log}(\boldsymbol{W}_j)} \tag{2.60}$$

The samples $\{\boldsymbol{W}_j\}_{j=1}^J$ can be generated by first generating a set of independent samples $\{\boldsymbol{U}_j\}_{j=1}^J$ from a standard normal distribution, and then substituting these samples into

$$\vec{\boldsymbol{W}}_j = \vec{\boldsymbol{W}}^* + \nabla^2_{\boldsymbol{W}} \mathcal{P}_\delta^{\log}(W^*)^{-1/2} \vec{\boldsymbol{U}}_j, \tag{2.61}$$

where $\nabla^2_{\boldsymbol{W}} \mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)^{-1/2}$ is defined from the Cholesky decomposition of the covariance matrix as in

$$\nabla^2_{\boldsymbol{W}} \mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)^{-1} = \nabla^2_{\boldsymbol{W}} \mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)^{-1/2} \nabla^2_{\boldsymbol{W}} \mathcal{P}_\delta^{\log}(\boldsymbol{W}^*)^{-\mathrm{T}/2}. \tag{2.62}$$

To summarize, we can estimate δ by minimizing the ABIC criterion in (2.57). In practice, this is done by calculating the criterion for various values of δ, followed by selecting the δ value giving the lowest criterion.

2.1.2 Garbage Class

In some applications, the classifier will be presented with observations $\boldsymbol{X}$ that do not correspond to any of the classes in the label set $\mathcal{Y}$. As an example, consider a classifier designed for classifying handwritten digits which is presented with a letter from the English alphabet. In this situation, the classifier should return a small probability for every class in $\mathcal{Y}$. However, this is made impossible by the fact that the total probability should sum to 1, that is, $\sum_{y \in \mathcal{Y}} p(y|\boldsymbol{X}) = 1$. The solution to this problem is to introduce a new class $y = C + 1 \in \mathcal{Y}_0 = \mathcal{Y} \cup \{K + 1\}$, called a *garbage class*, that should get high conditional probability given observations that are unlikely for the classes in $\mathcal{Y}$, and small probability otherwise [Birkenes et al., 2007].

In order to train the parameters of the logistic regression model with such a garbage class, a set of observations labeled with a garbage label, or *garbage observations*, are needed. For applications with a low-dimensional observation set $\mathcal{X}$, these garbage observations can be drawn from a uniform distribution over $\mathcal{X}$. For many practical applications however, $\mathcal{X}$ has a very high dimensionality, so an unreasonably high number of samples must be drawn from the uniform distribution in order to achieve good performance. In such cases, prior knowledge of the nature or the generation of the possible garbage observations that the classifier will see during prediction is of great

value. For the example with handwritten digits, the garbage observations may be extracted from handwritten letters of the English alphabet. In Chapter 5 we will see how we can use N-best lists to generate garbage observations for continuous speech recognition.

2.1.3 Adaptive Regressor Parameters

To gain additional discriminative power it was proposed in [Birkenes et al., 2006a] to treat $\boldsymbol{\lambda}$ as a free parameter of the logistic regression model instead of a preset fixed hyperparameter. In this setting, the criterion function can be written

$$\mathcal{P}_{\delta}^{\log}(\boldsymbol{W}, \boldsymbol{\lambda}; \mathcal{D}) = -\sum_{n=1}^{N} \log p(y = y^{(n)} | \boldsymbol{X}^{(n)}, \boldsymbol{W}, \boldsymbol{\lambda}) + \frac{\delta}{2} \operatorname{trace} \boldsymbol{\Gamma} \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{W}, \quad (2.63)$$

which is the same as the criterion in (2.23), but with the dependency on $\boldsymbol{\lambda}$ shown explicitly. Note that also $\boldsymbol{\Sigma}$ depends on $\boldsymbol{\lambda}$ according to (2.18). The goal of parameter estimation is now to find the *pair* $(\boldsymbol{W}^*, \boldsymbol{\lambda}^*)$ that minimizes the criterion in (2.63). This can be written mathematically as

$$(\boldsymbol{W}^*, \boldsymbol{\lambda}^*) = \arg \min_{(\boldsymbol{W}, \boldsymbol{\lambda})} \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}, \boldsymbol{\lambda}; \mathcal{D}). \quad (2.64)$$

As already mentioned, the function in (2.63) is convex with respect to $\boldsymbol{W}$ if $\boldsymbol{\lambda}$ is held fixed. It is not guaranteed, however, that it is convex with respect to $\boldsymbol{\lambda}$ if $\boldsymbol{W}$ is held fixed. Therefore, the best we can hope for is to find a local minimum that gives good classification performance.

A local minimum can be obtained by using a coordinate descent approach with coordinates $\boldsymbol{W}$ and $\boldsymbol{\lambda}$. The algorithm is initialized with $\boldsymbol{\lambda}_0$. Then the initial weight matrix can be found as

$$\boldsymbol{W}_0 = \arg \min_{\boldsymbol{W}} \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}, \boldsymbol{\lambda}_0; \mathcal{D}). \quad (2.65)$$

The iteration step is as follows:

$$\boldsymbol{\lambda}_{i+1} = \arg \min_{\boldsymbol{\lambda}} \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}_i, \boldsymbol{\lambda}; \mathcal{D}), \quad (2.66a)$$

$$\boldsymbol{W}_{i+1} = \arg \min_{\boldsymbol{W}} \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}, \boldsymbol{\lambda}_{i+1}; \mathcal{D}). \quad (2.66b)$$

The coordinate descent method is illustrated in Figure 2.3.

For the convex minimization with respect to $\boldsymbol{W}$, we can use the penalized logistic regression machine (PLRM) in Algorithm 2.1.3. As for the

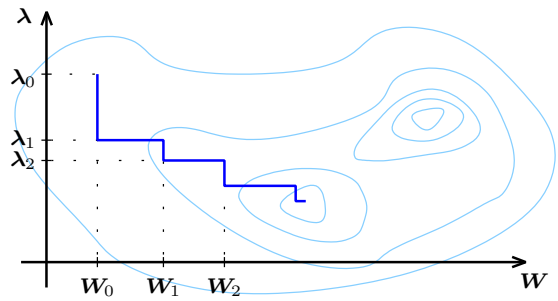

Figure 2.3: The coordinate descent method used to find the pair $(\boldsymbol{W}^*, \boldsymbol{\lambda}^*)$ that minimizes the criterion function $\mathcal{P}_\delta^{\log}(\boldsymbol{W}, \boldsymbol{\lambda}; \mathcal{D})$.

minimization with respect to $\boldsymbol{\lambda}$, there are many possibilities, two of which are the steepest descent method and the RProp method [Riedmiller and Braun, 1993]. Both these optimization methods make use of the partial derivatives of the criterion function with respect to each of the elements of the L-dimensional vector $\boldsymbol{\lambda}$. The partial derivative of the criterion function with respect to λ_l is given in the following lemma.

Lemma 2.1.4 *The partial derivative of* (2.63) *with respect to the lth element of* $\boldsymbol{\lambda}$ *is*

$$\frac{\partial}{\partial \lambda_l} \mathcal{P}_\delta^{\log}(\boldsymbol{W}; \boldsymbol{\lambda}; \mathcal{D}) = \operatorname{trace}\left\{ \left(\frac{\delta}{N} \boldsymbol{\Phi}^T \boldsymbol{W} \boldsymbol{\Gamma} + \boldsymbol{P}^{\mathrm{T}}(\boldsymbol{W}) - \boldsymbol{Y}^{\mathrm{T}} \right) \boldsymbol{W}^{\mathrm{T}} \frac{\partial}{\partial \lambda_l} \boldsymbol{\Phi} \right\}. \tag{2.67}$$

Proof See App. A

When the regressor parameters are updated by minimizing the criterion function in (2.63), overfitting to the training data may occur. This typically happens when the number of free parameters in the regressor functions is large compared to the available training data. This issue is known as the *curse of dimensionality*. By keeping the number of free parameters in accordance with the number of training examples, the effect of overfitting may be reduced.

Another method to reduce the effect of overfitting is *early stopping*. In this method, a part of the training set, known as a *validation set*, is used to monitor the generalization performance, i.e., the recognition accuracy on data not seen by the training algorithm, as the training algorithm iterates. Training is stopped when the generalization performance reaches a maximum. Early stopping reduces the effect of overfitting by ensuring that the parameters do not deviate too much from their initial values.

A third way to reduce the effect of overfitting is to add a penalty term for the regressor parameters to the criterion function in (2.63). Let us assume that each element λ_i of the L-dimensional vector $\boldsymbol{\lambda}$ has a Gaussian prior with mean μ_i and variance σ_i^2 that are known. Then, a penalty term for the whole vector can be written

$$\pi'(\boldsymbol{\lambda}) = \prod_{i=1}^{L} e^{-\frac{1}{2\sigma^2}(\lambda_i - \mu_i)^2} \tag{2.68}$$

$$= e^{-\frac{1}{2}\sum_{i=1}^{L} \frac{1}{\sigma_i^2}(\lambda_i - \mu_i)^2} \tag{2.69}$$

$$= e^{-\frac{1}{2}(\boldsymbol{\lambda}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}'(\boldsymbol{\lambda}-\boldsymbol{\mu})}, \tag{2.70}$$

where $\boldsymbol{\mu}$ is a vector of all means μ_i, and $\boldsymbol{\Sigma}'$ is a diagonal matrix with elements $1/\sigma_i^2$. The criterion function including a penalty for the regressor parameters is then

$$\mathcal{P}^{\log}_{\delta,\delta'}(\boldsymbol{W}, \boldsymbol{\lambda}; \mathcal{D}) = -\sum_{n=1}^{N} \log p(y = y^{(n)} | \boldsymbol{X}^{(n)}, \boldsymbol{W}, \boldsymbol{\lambda}) + \frac{\delta}{2} \operatorname{trace} \boldsymbol{\Gamma} \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{W} + \frac{\delta'}{2} (\boldsymbol{\lambda} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}' (\boldsymbol{\lambda} - \boldsymbol{\mu}), \tag{2.71}$$

where δ' is a regularization parameter for the regressor parameters.

2.2 Kernel Logistic Regression

Kernel logistic regression (KLR) is a generalization of penalized logistic regression. It allows for more flexible decision boundaries by letting the number of regressors grow to a very large number, or even infinite, without much increase in computational cost. Moreover, in KLR, we are not required to make specific choices of the mapping $\boldsymbol{\phi}$ and the matrix $\boldsymbol{\Sigma}$. These quantities are implicitly defined through the *kernel*, which is a function that can be thought of as a similarity measure between pairs of observations, and which plays an important role in any kernel method. The following presentation of KLR is largely based on [Tanabe, 2001a,b] and [Tanabe, 2003].

In the previous section we saw that the minimizer $\boldsymbol{W}^*$ of the negative logarithm of the penalized likelihood satisfies

$$\boldsymbol{W}^* = \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi} \boldsymbol{V}^*, \tag{2.72}$$

where $\boldsymbol{V}^* = 1/\delta(\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{P}(\boldsymbol{W}^*)^{\mathrm{T}})\boldsymbol{\Gamma}^{-1}$. The $N \times C$ matrix $\boldsymbol{V}$ defined in $\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\boldsymbol{V}$ is called the *dual parameter matrix*. Substituting this into

the criterion function in (2.23) gives

$$\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \mathcal{P}_\delta^{\log}(\boldsymbol{W};\mathcal{D}) \tag{2.73}$$

$$= \mathcal{P}_\delta^{\log}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\boldsymbol{V};\mathcal{D}) \tag{2.74}$$

$$= -\sum_{n=1}^{N} \log p(y = y^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{V}) + \frac{\delta}{2}\operatorname{trace}\boldsymbol{\Gamma}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}, \tag{2.75}$$

with

$$p(y = y^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{V}) = \frac{e^{\boldsymbol{v}_{y^{(n)}}^{\mathrm{T}}\boldsymbol{k}(\boldsymbol{X}^{(n)})}}{\sum_{j=1}^{C} e^{\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{k}(\boldsymbol{X}^{(n)})}}, \tag{2.76}$$

where the *kernel matrix* $\boldsymbol{K} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}$ is an $(N \times N)$-dimensional matrix with columns $\boldsymbol{k}(\boldsymbol{X}^{(n)}) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})$, and $\boldsymbol{v}_i$ denotes the columns of $\boldsymbol{V}$. The kernel matrix can be written as

$$\boldsymbol{K} = \begin{bmatrix} | & | & & | \\ \boldsymbol{k}(\boldsymbol{X}^{(1)}) & \boldsymbol{k}(\boldsymbol{X}^{(2)}) & \dots & \boldsymbol{k}(\boldsymbol{X}^{(N)}) \\ | & | & & | \end{bmatrix} \tag{2.77}$$

$$= \begin{bmatrix} k(\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(1)}) & k(\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}) & \dots & k(\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(N)}) \\ k(\boldsymbol{X}^{(2)}, \boldsymbol{X}^{(1)}) & k(\boldsymbol{X}^{(2)}, \boldsymbol{X}^{(2)}) & & \\ \vdots & & \ddots & \\ k(\boldsymbol{X}^{(N)}, \boldsymbol{X}^{(1)}) & & & k(\boldsymbol{X}^{(N)}, \boldsymbol{X}^{(N)}) \end{bmatrix}, \tag{2.78}$$

where each element has the quadratic form

$$k(\boldsymbol{X}, \boldsymbol{X}') = \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X};\boldsymbol{\lambda})\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}(\boldsymbol{X}';\boldsymbol{\lambda}) \tag{2.79}$$

for some $\boldsymbol{X}, \boldsymbol{X}' \in \mathcal{X}$. The function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel* and computes a real number $k(\boldsymbol{X}, \boldsymbol{X}')$ from two features $\boldsymbol{X}$ and $\boldsymbol{X}'$. The kernel is of central importance in kernel logistic regression and will be discussed in more detail later.

Similar to (2.76), the conditional probability of y given a new feature $\boldsymbol{X}$ using the kernel logistic regression model can be expressed in terms of the dual parameters and kernels as

$$p(y = i|\boldsymbol{X}, \boldsymbol{V}) = \frac{e^{\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{k}(\boldsymbol{X})}}{\sum_{j=1}^{C} e^{\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{k}(\boldsymbol{X})}} \quad \text{for } i = 1, \dots, C, \tag{2.80}$$

where $\boldsymbol{k}(\boldsymbol{X}) = [k(\boldsymbol{X}^{(1)}, \boldsymbol{X}), \dots, k(\boldsymbol{X}^{(N)}, \boldsymbol{X})]^{\mathrm{T}}$ is an N-dimensional vector consisting of the kernels computed between $\boldsymbol{X}$ and each training feature $\boldsymbol{X}^{(n)}$. The kernel logistic regression model is illustrated in Figure 2.4.

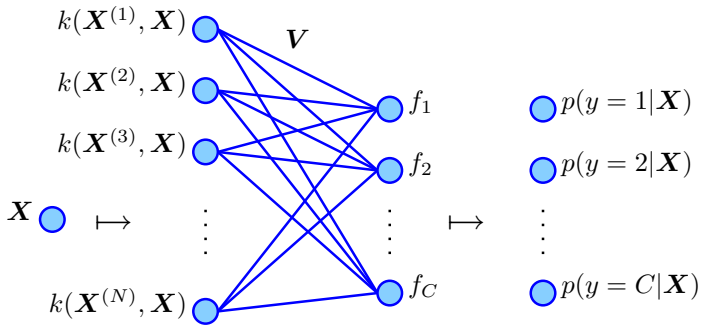

Figure 2.4: The kernel logistic regression model.

From equations (2.75), (2.76) and (2.80) above, we note that there are no explicit reference neither to the map $\boldsymbol{\phi}$ nor the positive definite matrix $\boldsymbol{\Sigma}$. The training of the kernel logistic regression model through minimization of the criterion $\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}; \mathcal{D})$ in (2.75) can be carried out only with the quantities $\boldsymbol{K}$, $\boldsymbol{\Gamma}$, δ, and the training set labels $\{y^{(1)}, \ldots, y^{(N)}\}$. In the prediction of a new feature $\boldsymbol{X}$, the weight matrix $\boldsymbol{V}$ and the vector $\boldsymbol{k}(\boldsymbol{X})$ of computed kernels between $\boldsymbol{X}$ and each training feature are needed. Note that the dimension that dominates the quantities needed in kernel logistic regression is the number of training examples N, irrespective of the dimension of the underlying mapping $\boldsymbol{\phi}$. In practice, this allows for a very large dimension of the image of $\boldsymbol{\phi}$ as long as the kernel k can be efficiently computed. We will see in the next subsection that many such kernels and corresponding mappings exist.

In the following we will be concerned with the minimization of the criterion $\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}; \mathcal{D})$ in (2.75) with respect to the dual parameter matrix $\boldsymbol{V}$. An optimization algorithm will be presented that makes use of both the gradient and the Hessian of the criterion function, whose expressions are given in the following two lemmas.

Lemma 2.2.1 *The gradient of* (2.75) *is the* $N \times C$ *matrix*

$$\nabla_{\boldsymbol{V}} \check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}; \mathcal{D}) = \boldsymbol{K}(\check{\boldsymbol{P}}(\boldsymbol{V})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}} + \delta \boldsymbol{V} \boldsymbol{\Gamma}), \tag{2.81}$$

where $\check{\boldsymbol{P}}(\boldsymbol{V}) = \boldsymbol{P}(\boldsymbol{W}) = \boldsymbol{P}(\boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi} \boldsymbol{V})$.

Proof See App. A

Lemma 2.2.2 *The Hessian of* (2.75) *is*

$$\nabla_{\boldsymbol{V}}^2 \check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}; \mathcal{D}) = \sum_{n=1}^{N} (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{X}^{(n)}) + \delta \boldsymbol{\Gamma} \otimes \boldsymbol{K}. \tag{2.82}$$

Proof See App. A

Since $\mathcal{P}_\delta^{\log}$ is convex with respect to $\boldsymbol{W}$, and $\boldsymbol{V}$ is a linear transformation of $\boldsymbol{W}$, the criterion function $\check{\mathcal{P}}_\delta^{\log}$ is convex with respect to $\boldsymbol{V}$ with a unique minimum that occurs for the minimizer

$$\boldsymbol{V}^* = \frac{1}{\delta}(\boldsymbol{Y}^{\mathrm{T}} - \check{\boldsymbol{P}}(\boldsymbol{V}^*)^{\mathrm{T}})\boldsymbol{\Gamma}^{-1}. \tag{2.83}$$

The above equation is the result of setting the gradient in (2.81) to zero. Note that $\boldsymbol{V}^*$ in the equation above is the same as the one in (2.72), and hence the minimum obtained with $\boldsymbol{V}^*$ is exactly the same as the minimum obtained with $\boldsymbol{W}^*$. This means that we can obtain the same result by optimizing the criterion function with respect to the dual parameter matrix $\boldsymbol{V}$ instead of the primal parameter matrix $\boldsymbol{W}$.

We will here present an optimization algorithm for estimating the weight matrix $\boldsymbol{V}$ that was introduced in [Tanabe, 2001a] where it was called the *dual penalized logistic regression machine* (dPLRM). In this algorithm, the weight matrix is updated iteratively using Newton's method, where each step is

$$\boldsymbol{V}^{i+1} = \boldsymbol{V}^i - \alpha_i \Delta \boldsymbol{V}^i, \tag{2.84}$$

where $\Delta \boldsymbol{V}^i$ is defined in

$$\operatorname{vec} \Delta \boldsymbol{V}^i = [\nabla^2_{\boldsymbol{V}} \check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}^i; \mathcal{D})]^{-1} \operatorname{vec} \nabla_{\boldsymbol{V}} \check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}^i; \mathcal{D}). \tag{2.85}$$

As for the optimization of the penalized likelihood that was presented in the previous section, we compute an approximation to $\Delta \boldsymbol{V}^i$ using the conjugate gradient (CG) method, since the inverse of the Hessian matrix is costly to obtain. This amounts to solving for $\Delta \boldsymbol{V}^i$ in the equation

$$\nabla^2_{\boldsymbol{V}} \check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}^i; \mathcal{D}) \operatorname{vec} \Delta \boldsymbol{V}^i = \operatorname{vec} \nabla_{\boldsymbol{V}} \check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}^i; \mathcal{D}), \tag{2.86}$$

which after substitution of (2.81) and (2.82) reduces to

$$\sum_{n=1}^{N} \boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{X}^{(n)}) \Delta \boldsymbol{V}^i (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) + \delta \boldsymbol{K} \Delta \boldsymbol{V}^i \boldsymbol{\Gamma}$$
$$= \boldsymbol{K}(\check{\boldsymbol{P}}(\boldsymbol{V}^{i\mathrm{T}}) - \boldsymbol{Y}^{\mathrm{T}} + \delta \boldsymbol{V}^i \boldsymbol{\Gamma}). \tag{2.87}$$

If $\boldsymbol{K}$ is non-singular, we can pre-multiply both sides of the above equation with $\boldsymbol{K}^{-1}$, which yields

$$\sum_{n=1}^{N} \boldsymbol{e}_n \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{X}^{(n)}) \Delta \boldsymbol{V}^i (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) + \delta \Delta \boldsymbol{V}^i \boldsymbol{\Gamma}$$
$$= \check{\boldsymbol{P}}(\boldsymbol{V}^{i\mathrm{T}}) - \boldsymbol{Y}^{\mathrm{T}} + \delta \boldsymbol{V}^i \boldsymbol{\Gamma} \tag{2.88}$$

since $\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{X}^{(n)}) = \boldsymbol{e}_n$, where $\boldsymbol{e}_n$ is the unit vector with all zeros except for the nth element which is 1.

The CG method for solving this equation is summarized in the following algorithm, which computes an estimate of the weight matrix $\boldsymbol{V}$ that minimizes the criterion $\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V}; \mathcal{D})$ in (2.75).

Algorithm 2.2.3 (The dual penalized logistic regression machine [Tanabe, 2003]) *Start with an initial weight matrix $\boldsymbol{V}_0$ and generate a sequence of matrices according to*

$$\boldsymbol{V}^{i+1} = \boldsymbol{V}^i - \alpha_i \Delta \boldsymbol{V}^i, \tag{2.89}$$

where $\Delta\boldsymbol{V}^i$ is computed using the following conjugate gradient method:

1. *Initialize: Start with an initial matrix $\Delta\boldsymbol{V}_0^i$ and compute the matrices $\boldsymbol{R}_0$ and $\boldsymbol{Q}_0$:*

$$\begin{aligned}\boldsymbol{R}_0 &= \check{\boldsymbol{P}}(\boldsymbol{V}^{i\mathrm{T}}) - \boldsymbol{Y}^{\mathrm{T}} + \delta \boldsymbol{V}^i \boldsymbol{\Gamma} \\ &\quad - \sum_{n=1}^{N} \boldsymbol{e}_n \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{X}^{(n)}) \Delta \boldsymbol{V}_0^i (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) - \delta \Delta \boldsymbol{V}_0^i \boldsymbol{\Gamma},\end{aligned} \tag{2.90}$$

$$\boldsymbol{Q}_0 = \boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{e}_n^{\mathrm{T}} \boldsymbol{R}_0 (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) - \boldsymbol{R}_0 \boldsymbol{\Gamma}. \tag{2.91}$$

2. *Iterate: Generate a sequence $(\Delta\boldsymbol{V}_1^i, \Delta\boldsymbol{V}_2^i, \dots)$ according to*

$$\alpha_j = \frac{\|\boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{e}_n^{\mathrm{T}} \boldsymbol{R}_j (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) + \boldsymbol{R}_j \boldsymbol{\Gamma}\|^2}{\|\boldsymbol{e}_n \boldsymbol{k}(\boldsymbol{X}^{(n)})^{\mathrm{T}} \boldsymbol{Q}_j (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) + \boldsymbol{Q}_j \boldsymbol{\Gamma}\|^2}, \tag{2.92}$$

$$\Delta \boldsymbol{V}_{j+1}^i = \Delta \boldsymbol{V}_j^i + \alpha_j \boldsymbol{Q}_j, \tag{2.93}$$

$$\begin{aligned}\boldsymbol{R}_{j+1} &= \check{\boldsymbol{P}}(\boldsymbol{V}^{i\mathrm{T}}) - \boldsymbol{Y}^{\mathrm{T}} + \delta \boldsymbol{V}^i \boldsymbol{\Gamma} \\ &\quad - \sum_{n=1}^{N} \boldsymbol{e}_n \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{X}^{(n)}) \Delta \boldsymbol{V}_{j+1}^i (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) - \delta \Delta \boldsymbol{V}_{j+1}^i \boldsymbol{\Gamma},\end{aligned} \tag{2.94}$$

$$\beta_{j+1} = \frac{\|\boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{e}_n^{\mathrm{T}} \boldsymbol{R}_{j+1} (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) + \boldsymbol{R}_{j+1} \boldsymbol{\Gamma}\|^2}{\|\boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{e}_n^{\mathrm{T}} \boldsymbol{R}_j (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) + \boldsymbol{R}_j \boldsymbol{\Gamma}\|^2}, \tag{2.95}$$

$$\boldsymbol{Q}_{j+1} = \boldsymbol{k}(\boldsymbol{X}^{(n)}) \boldsymbol{e}_n^{\mathrm{T}} \boldsymbol{R}_{j+1} (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) - \boldsymbol{R}_{j+1} \boldsymbol{\Gamma} + \beta_{j+1} \boldsymbol{Q}_j. \tag{2.96}$$

2.2.1 The Kernel

A *kernel* is a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In kernel logistic regression and other kernel methods, not all kernels are interesting. We focus our attention on kernels that are *positive definite* or *conditionally positive definite*, as given in the following definition.

Definition [Schölkopf and Smola, 2002] Let $\mathcal{X}$ be a nonempty set. A real-valued symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *positive definite kernel* if for all $N \in \mathbb{N}$ and all $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)} \in \mathcal{X}$ the induced $N \times N$ kernel matrix $\boldsymbol{K}$ with elements $k(\boldsymbol{X}^{(m)}, \boldsymbol{X}^{(n)})$ satisfies $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{a} \geq 0$ given any vector $\boldsymbol{a} \in \mathbb{R}^N$. The function k is called a *conditionally positive definite kernel* if $\boldsymbol{K}$ satisfies the above inequality for any vector $\boldsymbol{a} \in \mathbb{R}^N$ with $\sum_{n=1}^{N} a_n = 0$.

The following lemma relates the set of positive definite kernels and the set of conditionally positive definite kernels. We will be needing the lemma in Chapter 4 when we introduce a kernel for time series.

Lemma 2.2.4 *([Schölkopf and Smola, 2002])*

(i) Any positive definite kernel is also a conditionally positive definite kernel.

(ii) $e^{\beta k}$ is positive definite for all $\beta > 0$ if and only if k is conditionally positive definite.

Proof See [Schölkopf and Smola, 2002].

The kernel defined in (2.79) is positive definite since we require $\boldsymbol{\Sigma}$, and thereby $\boldsymbol{\Sigma}^{-1}$, to be a positive definite matrix. Note that this kernel can be written as an inner product between two vectors $\boldsymbol{\psi}(\boldsymbol{X})$ and $\boldsymbol{\psi}(\boldsymbol{X}')$ by letting $\boldsymbol{\psi}(\boldsymbol{X}) = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda})$. That is,

$$k(\boldsymbol{X}, \boldsymbol{X}') = \boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{X})\boldsymbol{\psi}(\boldsymbol{X}'), \tag{2.97}$$

where the mapping $\boldsymbol{\psi} : \mathcal{X} \to \mathbb{R}^{M+1}$ is defined in terms of the mapping $\boldsymbol{\phi}$ and the positive definite matrix $\boldsymbol{\Sigma}$. Now, it follows from Mercer's Theorem [Mercer, 1909; Schölkopf and Smola, 2002] that *any* positive definite kernel admits the form of an inner product $\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{X})\boldsymbol{\psi}(\boldsymbol{X}')$ for some mapping $\boldsymbol{\psi}$. Therefore, instead of choosing $\boldsymbol{\psi}$ explicitly through choices of $\boldsymbol{\phi}$ and $\boldsymbol{\Sigma}$, we could choose a positive definite kernel k that implicitly defines a mapping $\boldsymbol{\psi}$, which in turn implicitly defines $\boldsymbol{\phi}$ and $\boldsymbol{\Sigma}$. Then, with the choice of a kernel k, the dual parameter matrix $\boldsymbol{V}$ can be estimated by

minimizing the criterion $\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D})$ in (2.75), and the conditional probability $p(y=i|\boldsymbol{X},\boldsymbol{V})$ in (2.80) can be computed, without explicitly defining $\boldsymbol{\phi}$ and $\boldsymbol{\Sigma}$. Herein lays the elegance and power of kernel logistic regression, and more generally kernel methods. By choosing a positive definite kernel k, there exists a mapping $\boldsymbol{\psi}$ that maps the features into a possible infinite dimensional space, where linear prediction is performed. Moreover, depending on the choice of kernel, learning algorithms can run on a computer in a small amount of time and generate complex nonlinear classifiers that would be extremely difficult or even impossible using only the primal method of logistic regression.

As an example, let the observation space be $\mathcal{X}=\mathbb{R}^D$ and consider the *polynomial kernel*

$$k(\boldsymbol{x},\boldsymbol{x}')=(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'+1)^d, \tag{2.98}$$

where $\boldsymbol{x},\boldsymbol{x}'\in\mathbb{R}^D$ and d is a positive integer. The corresponding mapping ψ maps each vector $\boldsymbol{x}$ into the space of all monomials up to degree d of its elements. This is the space where linear prediction is performed. The inner product of two mapped vectors is efficiently computed through the kernel in (2.98).

Another example of a kernel for $\mathcal{X}=\mathbb{R}^D$ is the *Gaussian kernel*

$$k(\boldsymbol{x},\boldsymbol{x}')=e^{-\frac{||\boldsymbol{x}-\boldsymbol{x}'||^2}{2\sigma^2}}, \tag{2.99}$$

where $\boldsymbol{x},\boldsymbol{x}'\in\mathbb{R}^D$ and $\sigma>0$. The image of the associated mapping ψ has infinite dimension.

The design of kernels for more complex sets $\mathcal{X}$ is an important research topic. Two general design methodologies are 1) to construct a kernel from generative models (e.g., [Jaakkola and Haussler, 1999a]), and 2) to construct a kernel from a similarity measure (e.g., [Vert et al., 2004]). In Chapter 4 we will present various kernels for time series extracted from speech signals using both of the above design methodologies.

2.2.2 Sparse Approximations

In KLR, we need to store all the N inputs $\boldsymbol{X}^{(1)},\ldots,\boldsymbol{X}^{(N)}$, since they are needed in the classification of a new example $\boldsymbol{X}$ according to (2.80). We also need to store the $(N\times C)$-dimensional dual parameter matrix $\boldsymbol{V}$. If N is large, which is the case for many practical problems, the memory requirements and computational complexity may become impractically large. In many applications in automatic speech recognition, for example, the number of training examples can be several hundreds of thousands.

A way to overcome the above problem is to select only a small representative subset $\mathcal{S}$ of the training data for inclusion in the kernel logistic

regression model. In [Zhu and Hastie, 2001, 2005; Myrvoll and Matsui, 2006], the authors presented a greedy training algorithm for KLR. The algorithm is greedy in the sense that it starts with an empty set $\mathcal{S}$, and incrementally adds additional training examples to $\mathcal{S}$ based on which examples improve the criterion function the most. The subset is increased until convergence. The size of the obtained subset is typically smaller than the number of support vectors in the support vector machine (SVM) [Zhu and Hastie, 2001, 2005].

The approach taken in [Krishnapuram et al., 2005] is to use a sparse-promoting Laplacian prior instead of a Gaussian prior typically assumed for KLR. The price we have to pay for using a Laplace prior instead of a Gaussian prior, is a criterion function that is no longer differentiable. The authors in [Krishnapuram et al., 2005] propose to optimize a smooth bound on the criterion function instead of the original criterion function, in a similar fashion as the celebrated expectation maximization (EM) algorithm [Dempster et al., 1977]. The result is that many of the training examples will have zero weights and can therefore be omitted in the KLR model.

2.3 Summary

In this chapter, we presented the framework of logistic regression in the context of multiclass classification. Both penalized logistic regression (PLR) and kernel logistic regression (KLR) were considered. The logistic regression framework we presented is general in the sense that it can be applied to any kind of data. In particular, in the rest of this thesis we will consider the application of logistic regression to speech recognition, where the inputs are sequences of vectors. Two new concepts that we introduced were adaptive regressor parameters and garbage class. We will have more to say about these concepts in the context of speech recognition in the following chapters.

As a final note, we would like to make clear that KLR is very similar to Gaussian process (GP) classification [Williams and Barber, 1998; Jaakkola and Haussler, 1999b; Rasmussen and Williams, 2006]. The difference lies in that GP classification is a fully Bayesian approach, meaning that the posterior distribution of the parameters is used in prediction, while KLR only uses the MAP estimate of the parameters. The fully Bayesian approach has to deal with an analytically intractable integral that can only be evaluated numerically. Several methods exist, one of which uses the Laplace approximation [Williams and Barber, 1998].

Chapter 3

Speech Recognition and Hidden Markov Models

The hidden Markov model (HMM) is a powerful model for sequences of variable lengths. It has been well studied over the last few decades for automatic speech recognition applications. This chapter gives an overview of the HMM and its use in automatic speech recognition. We will make use of the HMM in subsequent chapters due to its strengths in sequence modeling.

There are generally two ways to explain the HMM. The first approach explains the HMM as a model for generating sequences of observations. The most prominent example of this approach is [Rabiner, 1989], which explains the HMM using a simple example concerning the generation of a sequence of colored balls from a set of urns each containing a different distribution of colored balls. The second approach is more probabilistic in the sense that it explains the HMM as a probability distribution over sequences. Examples of the latter approach are [Bilmes, 2006; Jordan, 2007]. The presentation in this chapter uses the latter approach.

We start by presenting the usual way of extracting a sequence of feature vectors from a speech signal. Then, in Section 3.2 we explain the HMM. Sections 3.3 and 3.4 concern the typical application of HMMs to isolated-word speech recognition and continuous speech recognition, respectively. Finally, Section 3.5 contains a short summary of the chapter.

3.1 Feature Extraction

In automatic speech recognition, it is common to extract a set of features from each speech signal. Classification is carried out on the set of features instead of the speech signals themselves. A good set of features should

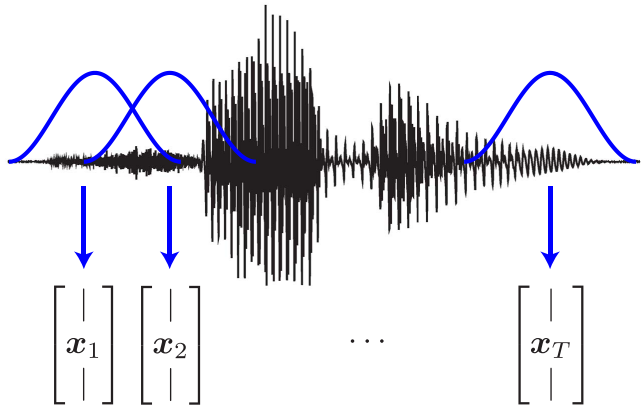

Figure 3.1: Feature extraction from a speech signal.

include discriminative information and exclude information that is irrelevant for classification (e.g., speaker dependent information such as pitch). Moreover, the feature set should be small enough to allow fast processing and robust.

A speech signal can be considered to be a realization of a *short-time stationary* stochastic process. This means that although the statistical characteristics of a speech signal change over time, they can be considered to be stationary within small time intervals (20-30 ms). This observation, together with the observation that the most prominent discriminative information between speech signals appear in the frequency domain, has lead to the common approach of extracting a time series which is a sequence of short time spectral feature vectors from each speech signal.

Figure 3.1 illustrates the extraction of a time series $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ from a speech signal. A window function of fixed width (often a Hamming window of width 20-30 ms) is used to confine processing to a short-time segment of the speech signal in order to generate a spectral feature vector. The window function is shifted a fixed length in time (typically 5-10 ms) to the right for further extraction of feature vectors until the end of the speech signal is reached.

Note that since different speech signals have different durations, feature extraction with a fixed window shift leads to time series with different number of vectors. This is one of the reasons why classification of speech signals is a more challenging task than classification of data whose features are merely fixed-dimensional vectors, which is the main concern of most classification methods in the literature.

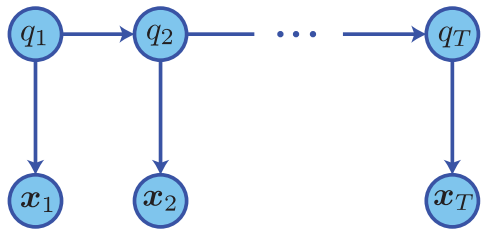

Figure 3.2: The hidden Markov model.

Popular feature vectors include linear prediction coefficients (LPC) and Mel-frequency cepstral coefficients (MFCC) [Davis and Mermelstein, 1980]. Often, the energy of the windowed speech signal is appended to the LPC or MFCC feature vectors. The dimension of these feature vectors is then typically 13.

It is well known that the inclusion of the *delta* and *acceleration coefficients* in the feature vectors can improve recognition performance considerably [Furui, 1986], especially when HMMs are used to model the feature vector sequences. Delta coefficients can be thought of as approximations to the first order time derivatives of a feature vector sequence, and contain information on the rate of change of the vectors in the sequence. Similarly, acceleration coefficients can be thought of as approximations to the second order time derivatives, and contain information on the rate of the rate of change. One delta and one acceleration coefficient are usually computed for each of the spectral coefficients and for the possibly appended energy feature. The typical dimension of a feature vector after appending the delta and acceleration coefficients is thus 39.

3.2 Hidden Markov Models

A *hidden Markov model* (HMM) is a model of the joint probability distribution $p(\boldsymbol{X}, \boldsymbol{q})$, where $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ is a time series and $\boldsymbol{q} = (q_1, \ldots, q_T)$ is a sequence of state variables q_t taking values in a finite set $\{1, \ldots, Q\}$ of states. Usually, the HMM is used to model time series $\boldsymbol{X}$, in which case the state variables q_t are latent or *hidden* variables. The probability distribution over time series can be found by summing the joint distribution $p(\boldsymbol{X}, \boldsymbol{q})$ over $\boldsymbol{q}$, i.e.,

$$p(\boldsymbol{X}) = \sum_{\boldsymbol{q}} p(\boldsymbol{X}, \boldsymbol{q}) \tag{3.1}$$

$$= \sum_{q_1} \cdots \sum_{q_T} p(\boldsymbol{X}, \boldsymbol{q}). \tag{3.2}$$

This computation seems at first daunting, especially for large T, but we will see shortly how it can be efficiently computed using the *forward algorithm* by exploiting the conditional independencies encoded in the HMM.

Figure 3.2 illustrates the HMM as a *graphical model* [Jordan, 2007]. From this graph, the main conditional independencies of interest can be identified as:

1. Conditioned on an arbitrary state variable, any preceding state variable is independent of any state variable that comes after, i.e.,

$$p(q_s, q_u|q_t) = p(q_s|q_t)p(q_u|q_t), \quad \text{for } s < t < u. \tag{3.3}$$

2. Conditioned on an arbitrary state variable, any preceding vector is independent of the current vector or any vector that comes after, i.e.,

$$p(\boldsymbol{x}_s, \boldsymbol{x}_u|q_t) = p(\boldsymbol{x}_s|q_t)p(\boldsymbol{x}_u|q_t), \quad \text{for } s < t \leq u. \tag{3.4}$$

Also, from the graph in 3.2, we can see that the joint distribution can be factored as

$$p(\boldsymbol{X}, \boldsymbol{q}) = p(q_1)\prod_{t=2}^{T} p(q_t|q_{t-1})\prod_{t=1}^{T} p(\boldsymbol{x}_t|q_t) \tag{3.5}$$

$$= \pi_{q_1}\prod_{t=2}^{T} a_{q_{t-1},q_t}\prod_{t=1}^{T} p(\boldsymbol{x}_t|q_t), \tag{3.6}$$

where $\pi_{q_1} = p(q_1)$ is the *initial state probability* and $a_{q_{t-1},q_t} = p(q_t|q_{t-1})$ is the *state transition probability* from state q_{t-1} to state q_t. Since the number of states Q is finite, all initial state probabilities can be represented in a Q-dimensional vector

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 \\ \vdots \\ \pi_Q \end{bmatrix} \tag{3.7}$$

where $\pi_q = p(q_1 = q)$ for $q \in \{1, \ldots, Q\}$. Similarly, all state transition probabilities can be represented in a $Q \times Q$ matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1Q} \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{Q1} & & & a_{QQ} \end{bmatrix}, \tag{3.8}$$

where $a_{qr} = p(q_t = r|q_{t-1} = q)$ for $q, r \in \{1, \ldots, Q\}$ and $t \in 2, \ldots, T$.

Now that we have factored the joint distribution $p(\boldsymbol{X}, \boldsymbol{q})$ according to the conditional independencies encoded in the HMM, we are ready to see how the likelihood in (3.2) can be efficiently computed. The likelihood of a time series $\boldsymbol{X}$ with respect to an HMM is

$$p(\boldsymbol{X}) = \sum_{q_1} \cdots \sum_{q_T} \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1},q_t} \prod_{t=1}^{T} p(\boldsymbol{x}_t|q_t). \tag{3.9}$$

By observing that each factor depends only on one or two state variables, we can write

$$\begin{aligned} p(\boldsymbol{X}) = \sum_{q_T} p(\boldsymbol{x}_T|q_T) \sum_{q_{T-1}} a_{q_{T-1},q_T} p(\boldsymbol{x}_{T-1}|q_{T-1}) \cdots \\ \cdots \sum_{q_2} a_{q_2,q_3} p(\boldsymbol{x}_2|q_2) \sum_{q_1} a_{q_1,q_2} p(\boldsymbol{x}_1|q_1) \pi_{q_1}. \end{aligned} \tag{3.10}$$

This equation is the basis for the *Forward Algorithm.*

Algorithm 3.2.1 (The Forward Algorithm) *Computes the likelihood* $p(\boldsymbol{X})$ *of a time series* $\boldsymbol{X}$ *with respect to an HMM.*

1. *Initialize:*
$$\alpha(q_1) = p(\boldsymbol{x}_1|q_1)\pi_{q_1}, \quad q_1 = 1, \ldots, Q \tag{3.11}$$

2. *Iterate: for* $t = 2, \ldots, T$
$$\alpha(q_t) = p(\boldsymbol{x}_t|q_t) \sum_{q_{t-1}} a_{q_{t-1},q_t} \alpha(q_{t-1}), \quad q_t = 1, \ldots, Q \tag{3.12}$$

3. *Terminate:*
$$p(\boldsymbol{X}) = \sum_{q_T} \alpha(q_T) \tag{3.13}$$

For each time t, the *forward variable* $\alpha(q_t)$ is the joint probability of the partial time series $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ and the state variable q_t, i.e.,

$$\alpha(q_t) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, q_t), \quad q_t = 1, \ldots, Q. \tag{3.14}$$

The Forward Algorithm has its name because it proceeds forward in time. This means that the computation of the likelihood can start once the first vector $\boldsymbol{x}_1$ has been observed, and can continue in steps each time a new vector is observed. This is an important property in real-time processing of speech.

In many applications, such as segmentation of speech signals, it is important to find the most likely state sequence given a time series. This can be done with the *Viterbi algorithm* [Viterbi, 1983], which computes the maximum likelihood estimate of the state sequence, i.e.,

$$\hat{\boldsymbol{q}} = \arg\max_{\boldsymbol{q}} p(\boldsymbol{X}, \boldsymbol{q}) \tag{3.15}$$

$$= \arg\max_{q_1} \ldots \arg\max_{q_T} p(\boldsymbol{X}, \boldsymbol{q}). \tag{3.16}$$

As a byproduct, the Viterbi algorithm also computes an approximation to the likelihood of a time series with respect to an HMM, namely

$$\hat{p}(\boldsymbol{X}) = \max_{\boldsymbol{q}} p(\boldsymbol{X}, \boldsymbol{q}) \tag{3.17}$$

$$= \max_{q_1} \ldots \max_{q_T} p(\boldsymbol{X}, \boldsymbol{q}), \tag{3.18}$$

which is obtained from (3.2) by replacing the sum operators with max operators. The Viterbi algorithm is as follows.

Algorithm 3.2.2 (The Viterbi Algorithm) *Computes the maximum likelihood state sequence* $\hat{\boldsymbol{q}}$ *as well as an approximation* $\hat{p}(\boldsymbol{X})$ *to the likelihood* $p(\boldsymbol{X})$ *of a time series* $\boldsymbol{X}$ *with respect to an HMM.*

1. *Initialize:*

$$\delta(q_1) = p(\boldsymbol{x}_1|q_1)\pi_{q_1}, \quad q_1 = 1, \ldots, Q \tag{3.19}$$

2. *Iterate: for* $t = 2, \ldots, T$

$$\delta(q_t) = p(\boldsymbol{x}_t|q_t) \max_{q_{t-1}} a_{q_{t-1},q_t}\delta(q_{t-1}), \quad q_t = 1, \ldots, Q \tag{3.20}$$

$$\psi(q_t) = \arg\max_{q_{t-1}} a_{q_{t-1},q_t}\delta(q_{t-1}), \quad q_t = 1, \ldots, Q \tag{3.21}$$

3. *Terminate:*

$$\hat{p}(\boldsymbol{X}) = \max_{q_T} \delta(q_T) \tag{3.22}$$

$$\hat{q}_T = \arg\max_{q_T} \delta(q_T) \tag{3.23}$$

4. *Backtrack: for* $t = T-1, \ldots, 1$

$$\hat{q}_t = \psi(\hat{q}_{t+1}) \tag{3.24}$$

In the HMM framework, we are free to choose the form of the *state-conditional distributions* $p(\boldsymbol{x}_t|q_t = q)$ for $q \in 1, \ldots, Q$. These Q conditional distributions are usually chosen to be members of the same parametric family with parameters $\boldsymbol{\eta}_q$ for $q \in 1, \ldots, Q$. In automatic speech recognition, it is popular to use Gaussian mixture models (GMM) as state-conditional distributions. A Gaussian mixture model is a weighted sum of Gaussian probability distributions, i.e.,

$$p(\boldsymbol{x}_t|q_t = q, \boldsymbol{\eta}_q) = \sum_{h=1}^{H} c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh}, \boldsymbol{\Sigma}_{qh}), \tag{3.25}$$

where c_{qh} is the mixture weight for state q and mixture h with $\sum_{h=1}^{H} c_{qh} = 1$, and H is the number of mixture components. Each state q and mixture h is associated with a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{qh}, \boldsymbol{\Sigma}_{qh})$ with mean vector $\boldsymbol{\mu}_{qh}$ and variance-covariance matrix $\boldsymbol{\Sigma}_{qh}$. The parameter set of the above state-conditional distribution is $\boldsymbol{\eta}_q = \{(c_{qh}, \boldsymbol{\mu}_{qh}, \boldsymbol{\Sigma}_{qh})\}_{h=1}^{H}$. If $\boldsymbol{\eta} = (\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_Q)$ denote the parameters of all the state-conditional distributions, we see that the HMM is parameterized as $p(\boldsymbol{X}, \boldsymbol{q}) = p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda})$ with parameter set $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\eta})$.

The maximum likelihood estimate of the HMM parameters can be found by maximizing the likelihood, or equivalently, by maximizing the log-likelihood with respect to the HMM parameters. Mathematically, the maximum likelihood estimate using a single observation $\boldsymbol{X}$ is

$$\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}\in\Lambda} \log p(\boldsymbol{X}; \boldsymbol{\lambda}) \tag{3.26}$$

$$= \arg\max_{\boldsymbol{\lambda}\in\Lambda} \log \sum_{\boldsymbol{q}} p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}), \tag{3.27}$$

where Λ denotes the parameter space of the HMM, i.e., the set of all allowable values for $\boldsymbol{\lambda}$. The above maximization is difficult since the log-likelihood is the logarithm of a sum. Fortunately, there is a simple and elegant iterative procedure for the estimation of the HMM parameters known as the *EM algorithm* [Dempster et al., 1977; Bilmes, 1997; Jordan, 2007]. The interested reader is referred to Appendix B for the details.

3.2.1 Hidden Markov Models for Speech

In real applications of HMMs, decisions must be made on the number of states, the allowed state transitions, and the form of the state-conditional probability distributions. In the following, we consider the application of HMMs to model time series extracted from speech signals.

A speech signal can be considered to consist of small units called *phones*. A phone is an utterance of a phoneme, which in turn is defined as the

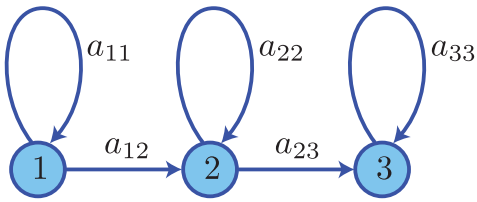

Figure 3.3: A left-to-right hidden Markov model with three states.

smallest linguistic unit that can carry meaning. In English, there are about 40 phonemes. Every word can be decomposed into a sequence of phonemes. As an example, the word "India" has the phoneme sequence "/ih/ /n/ /d/ /ih/ /ah/", where /ih/, /n/, /d/, /ih/ and /ah/ are phonemes from the English language. When HMMs are used to model speech, it is popular to use three HMM states for each phoneme in the phoneme sequence of a word; one state for the beginning of the phoneme, one state for the middle part, and one state for the end. Hence, the word "India", which consists of 5 consecutive phonemes would be modeled using 15 states.[1]

The order of the phonemes of a particular word is a feature that distinguishes it from other words. Therefore, it is common to order the states according to the phoneme order, start in the first state and allow only state transitions to the same state or the next state in the sequence. This is known as a *left-to-right* HMM and is realized by setting the initial state probability for the first state to one and the others to zero, and setting disallowed state-transition probabilities to zero. For example, for an HMM with three states, the initial state probability vector and the transition probability matrix are

$$\boldsymbol{\pi} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}, \tag{3.28}$$

respectively. Figure 3.3 illustrates a left-to-right HMM with three states. Note that this figure depicts the states and transition probabilities between states, in contrast to Figure 3.2, which depicts the state *variables*, the feature vectors, and their interdependencies.

As already mentioned, the class-conditional probability distributions are usually Gaussian mixture models as in (3.25). Often the variance-covariance matrices are assumed to be diagonal. This is a weak assumption when Mel-frequency cepstral coefficients (MFCC) are used as feature vectors because the different coefficients are uncorrelated by construction. The

[1] To be correct, we model the time series extracted from utterances of the word, and not the word itself.

main motivation for using diagonal variance-covariance matrices however, is to reduce the number of free parameters.

3.3 Isolated-Word Speech Recognition

Isolated-word speech recognition is the process of assigning a word to a given speech signal. The assigned word is an element of a fixed vocabulary and the speech signal is assumed to be an utterance of a word from the same vocabulary spoken in isolation. Although we use the term *word* in the present discussion, the theory is directly applicable to other linguistic units such as phonemes or sentences. Examples of isolated-word speech recognition include the classification of letters in the English alphabet, classification of phonemes, and classification of commands such as "yes", "no", "enter", etc.

In mathematical terms, let $\boldsymbol{X} \in \mathcal{X}$ denote a time series extracted from a speech signal, where $\mathcal{X}$ is the set of all such time series. Furthermore, let $y \in \mathcal{Y}$ denote a word label and $\mathcal{Y} = \{1, \ldots, C\}$ the set of word labels, where each word label corresponds to a word in the vocabulary. For example, in the recognition of letters in the English alphabet, $C = 26$, and $y = 1$ is the label for "A", $y = 2$ is the label for "B", etc. Speech recognition is a special case of classification, where the goal is to construct a decision rule $h : \mathcal{X} \to \mathcal{Y}$ that maps a time series $\boldsymbol{X} \in \mathcal{X}$ into a word label $\hat{y} \in \mathcal{Y}$.

The most popular approach to speech recognition is the generative approach that was presented in Chapter 1. Recall that the generative approach to classification amounts to estimating the distributions $p(\boldsymbol{X}|y)$ and $p(y)$, and substituting these estimates into the Bayes decision rule which is

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(\boldsymbol{X}|y)p(y). \tag{3.29}$$

A substitute for the class-conditional distribution $p(\boldsymbol{X}|y)$ for each word label $y \in \{1, \ldots, C\}$ is obtained from an HMM $p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}_i)$ with parameter set $\boldsymbol{\lambda}_i$ as in

$$p(\boldsymbol{X}|y = i) = p(\boldsymbol{X}; \boldsymbol{\lambda}_i) = \sum_{\boldsymbol{q}} p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}_i), \quad i = 1, \ldots, C. \tag{3.30}$$

The parameters are typically estimated from a set of training examples $\mathcal{D} = \{(\boldsymbol{X}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, y^{(N)})\}$ using the maximum likelihood method. For the prior probabilities $p(y)$, we can simply take the fraction of the training examples having the various word labels, or we can incorporate prior knowledge based on the speech recognition task at hand.

In recognizing a new speech signal with feature $\boldsymbol{X}$, we need to compute the likelihood with respect to each HMM, and decide on the word

label which gives the highest product of the HMM likelihood and the prior. For the likelihood computations, we can use for example the forward algorithm, or we can use the approximate likelihood computed by the Viterbi algorithm.

3.3.1 Discriminative Training of the HMM Parameters

It is well known that the HMM is not the correct model for time series extracted from speech signals. To see this, consider the conditional independence assumption in (3.4), which says that a feature vector is independent of the previous feature vector given the current state, i.e.,

$$p(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t|q_t) = p(\boldsymbol{x}_{t-1}|q_t)p(\boldsymbol{x}_t|q_t). \tag{3.31}$$

This is obviously a false assumption since the feature vectors $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$ are extracted from windowed speech segments that are either overlapping or at most some milliseconds apart. Nevertheless, the HMM is still the most popular model used in speech recognition.

Due to incorrect model assumptions and a finite amount of training data, the class-conditional probability distributions estimated using the maximum likelihood criterion may be far from the true unknown distributions. This will in turn lead to an error rate of the classifier that is considerably worse than the optimal error.

Instead of using the maximum likelihood criterion to estimate the HMM parameters, which optimizes each model independently of the others so as to best describe the training data, better recognition performance can be achieved by using a different criterion which optimizes all the models jointly and is more related to the goal of automatic speech recognition, namely the classification error rate. Such methods are called *discriminative training methods* and include both the *maximum mutual information* (MMI) method [Bahl et al., 1986] and the *minimum classification error* (MCE) method [Juang et al., 1997].

3.4 Continuous Speech Recognition

In the isolated-word speech recognition approach presented above, we are required to have at least one training example from each class since a distinct HMM has to be estimated for each class. This can be done without too much cost for systems involving a small number of classes such as the recognition of isolated digits. On the other hand, to collect training data for systems having a large number of classes would be either very expensive or practically infeasible. Consider for example the recognition of 8-digit

numbers, in which case the number of classes is $C = 10^8$. Or consider the recognition of freely spoken sentences, where the number of classes approaches infinity. It is clear that for such a large number of classes, we cannot take the isolated-word speech recognition approach in Section 3.3.

In the example above for the recognition of 8-digit numbers, an important observation to make is that every digit sequence is composed of only 10 subwords, namely the digits. Thus, by having one HMM per digit, we can form a model for any 8-digit number, i.e., any class, by concatenating the corresponding digit models. To take this a step further, we should note the fact that *any* spoken word or sentence is composed of a small number of subword units called *phonemes*. In English, there are about 40 phonemes, so with only about 40 HMMs, one for each phoneme, we can form a model for any English word or sentence. For the training of the phoneme HMMs, we only need to have one example of each phone, in principle. Of course, for good performance, this number should be much greater than one. The training is done with the Baum-Welch re-estimating formulae [Baum et al., 1970] on concatenated HMMs instead of word HMMs as in isolated-word speech recognition.

Once a set of subword models have been trained, a new utterance can be recognized by doing a Viterbi search for the most likely subword sequence weighted by the value of a language model. The search can be very computationally expensive, so various approximations have been proposed [Aubert, 2002]. The language model can be designed to the speech recognition problem at hand using expert human knowledge, or it can be a statistical N-gram model computed from a large text corpora [Rosenfeld, 2000]. In this thesis, a flat language model will be used. This means that all subword sequences are equally likely.

3.4.1 N-Best Lists and Lattices

Instead of generating only the most likely subword sequence, or sentence hypothesis, for an utterance, it is sometimes desirable to generate several competing hypotheses to be used in further processing. In this way, additional knowledge can be introduced before making a final decision. Two ways of representing a set of competing sentence hypotheses are *N-best lists* and *lattices*.

An N-best list [Schwartz and Chow, 1990] is a list of the N most likely sentence hypotheses of a given utterance, and can be efficiently generated from a set of HMMs. The sentence hypotheses are ordered by their HMM likelihood, and each hypothesis is accompanied by a segmentation, which is the most likely segment boundaries given the sentence.

A lattice can be generated from an N-best list by creating a directed

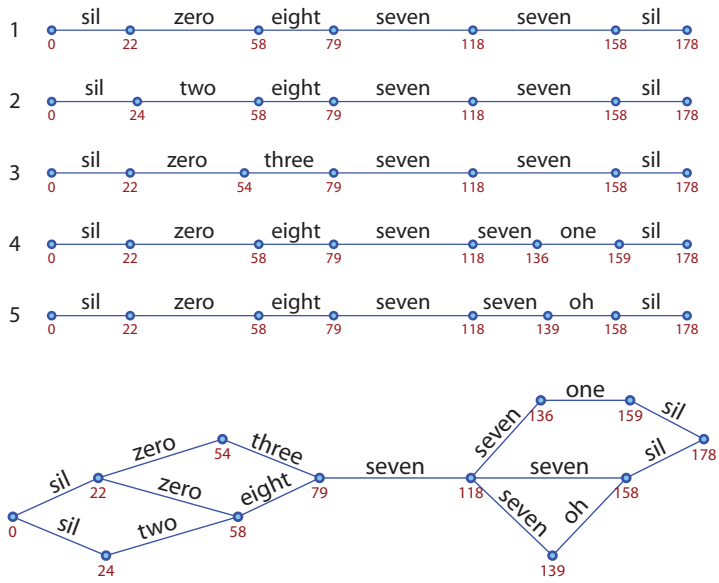

Figure 3.4: Top: An example of a 5-best list. The numbers under the nodes are frame numbers (in tens of milliseconds). Bottom: The lattice generated from the 5-best list above.

acyclic graph where each node corresponds to a time and each edge is a segment with corresponding labels. Each path from left to right in the lattice corresponds to a sentence hypothesis. It is easy to see that a lattice encodes more sentence hypotheses than the N-best list it was generated from.

Figure 3.4 shows an example of a 5-best list (top) obtained from an utterance containing a digit string, and the lattice (bottom) generated from the 5-best list. The numbers under the nodes are frame numbers (in tens of milliseconds). The lattice contains a total of 9 sentence hypotheses.

3.5 Summary

In this chapter we have given a brief introduction to the hidden Markov model (HMM) and its use in automatic speech recognition. We have chosen to present the HMM as a probability distribution over sequences, in the same manner as in [Bilmes, 2006; Jordan, 2007]. The HMM plays an important role in the subsequent chapters.

Chapter 4

Isolated-Word Speech Recognition using Logistic Regression

Isolated-word speech recognition is the task of recognizing words spoken in isolation. One of the main difficulties of this task is that speech signals typically have variable durations. In this chapter we present various mappings from variable length feature vector sequences into fixed-dimensional vectors. The mappings are either defined explicitly from a set of hidden Markov models (HMMs) to be used in penalized logistic regression (PLR), or implicitly through a kernel function to be used in kernel logistic regression (KLR). Notable examples of mappings and kernels that we will present are the *likelihood mapping*, the *likelihood-ratio mapping*, and the *global alignment (GA) kernel*. The mappings and kernels that we present can be used directly in the logistic regression framework to perform classification of speech signals containing words spoken in isolation.

For simplicity, we consider only the classification of a speech signal into a word. It should be clear, however, that the methods presented in this chapter are equally applicable to classification into other linguistic units such as phones. In Chapter 5 we will use one of the methods introduced in this chapter to perform classification of phone segments for the purpose of rescoring N-best lists for continuous speech recognition.

The outline of the chapter is as follows. First, we present mappings to be used with PLR and discuss the use of adaptive regressor parameters. Then, in Section 4.2 we present various kernels that can be used in KLR. Section 4.3 contains experimental results, and Section 4.4 contains a short summary and a discussion.

Part of the work presented in this chapter is based on [Birkenes et al.,

2005, 2006a; Cuturi et al., 2007].

4.1 Penalized Logistic Regression

In this section we will be concerned with the modeling of the conditional distribution $p(y|\boldsymbol{X})$ using the logistic regression model, where each observation $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ is a sequence of feature vectors extracted from a speech signal and y is a word label. Recall from Chapter 2 that the logistic regression model with parameter matrix $\boldsymbol{W}$ with columns $\boldsymbol{w}_c$ is

$$p(y = i|\boldsymbol{X}) = \frac{e^{\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}}{\sum_{j=1}^{C} e^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}} \quad \text{for } i = 1, \ldots, C. \tag{4.1}$$

Since the observation $\boldsymbol{X}$ is here a sequence of feature vectors that can vary in length, the mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^{M+1}$ is a map from the set $\mathcal{X}$ of all such observations into the Euclidean space $\mathbb{R}^{M+1}$. The hyperparameter of the mapping is $\boldsymbol{\lambda}$. Once a mapping $\boldsymbol{\phi}$ is specified, PLR for speech signals can be performed as in Chapter 2. In particular, classification of an observation $\boldsymbol{X}$ is accomplished by selecting the word having the largest conditional probability, that is,

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(y|\boldsymbol{X}), \tag{4.2}$$

where $\mathcal{Y}$ is the set of all allowable words known as the *vocabulary*.

The mapping should be able to map observations of varying lengths into fixed dimensional vectors while preserving the discriminative information embedded in the observations. There are many proposed methods on how to do this. Some methods select a fixed number of representative vectors from each feature vector sequence [Bazzi and Katabi, 2000; Ganapathiraju et al., 2004]. Other methods make use of one or more hidden Markov models (HMMs) to construct mappings, since the HMM framework is a principled way to handle variable length time series and have been well studied in the context of speech for a long time. With these mappings, the HMM parameters must be estimated before the parameter matrix $\boldsymbol{W}$ can be found. The HMM parameters can be estimated, for example, by using the maximum likelihood criterion or through discriminative training as explained in Chapter 3. After the weight matrix has been estimated, classification of a new observation $\boldsymbol{X}$ can be done as in (4.2).

In [Jaakkola and Haussler, 1999a], the authors suggested to map a feature vector sequence into the gradient space of the log-likelihood of a single HMM with respect to its parameters. This mapping is known as the *Fisher mapping* and can be written

$$\boldsymbol{\phi}_{\text{Fisher}}(\boldsymbol{X};\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{X};\boldsymbol{\lambda}), \tag{4.3}$$

where $p(\boldsymbol{X};\boldsymbol{\lambda})$ is the likelihood of a HMM with parameter $\boldsymbol{\lambda}$. The Fisher mapping is used to define the Fisher kernel that we will present in the next section.

In [Smith and Gales, 2002], the authors considered two-class classification problems with support vector machines (SVMs) using one or two HMMs for the mapping. In particular, they suggested a mapping from a sequence into a vector consisting of the log-likelihood-ratio with respect to two HMMs as well as the gradient of the log-likelihood ratio. If $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$ denote the HMM parameters of the two models, their mapping, which we call the *generative mapping*, can be expressed as

$$\boldsymbol{\phi}_{\text{Gen}}(\boldsymbol{X};\boldsymbol{\lambda}) = \begin{bmatrix} \log \frac{\tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(1)})}{\tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(2)})} \\ \nabla_{\boldsymbol{\lambda}^{(1)}} \log \tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(1)}) \\ -\nabla_{\boldsymbol{\lambda}^{(2)}} \log \tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(2)}) \end{bmatrix}, \tag{4.4}$$

where $\tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(1)})$ denotes a HMM likelihood which is normalized with respect to the sequence length according to [Smith and Gales, 2002]. The hyperparameter of the generative mapping is the pair $\boldsymbol{\lambda} = (\boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)})$ consisting of all the parameters of the two HMMs.

In the *conditional augmented (C-Aug) models* introduced in [Layton and Gales, 2006], there is one HMM with parameter $\boldsymbol{\lambda}^{(i)}$ for each class i. Each C-Aug model $p(y = i|\boldsymbol{X})$ incorporates the mapping

$$\boldsymbol{\phi}^{(i)}_{\text{C-Aug}}(\boldsymbol{X};\boldsymbol{\lambda}) = \begin{bmatrix} \log \tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(i)}) \\ \nabla^{(1,\rho)}_{\boldsymbol{\lambda}^{(i)}} \log \tilde{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(i)}) \end{bmatrix} \quad \text{for } i = 1, \ldots, C, \tag{4.5}$$

where the last element is a vector of the derivatives of order 1 to ρ of the log-likelihood with respect to the ith HMM. The hyperparameter of the class-conditional mappings in the C-Aug model is the set $\boldsymbol{\lambda} = (\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(C)})$

The authors in [Abou-Moustafa et al., 2004] presented a mapping that makes use of $M = C$ HMMs, one for each word in the vocabulary. The ith element of the mapped vector is the log-likelihood of the observation $\boldsymbol{X}$ with respect to the ith HMM, i.e.,

$$\boldsymbol{\phi}_{\text{Abou}}(\boldsymbol{X};\boldsymbol{\lambda}) = \begin{bmatrix} \log p(\boldsymbol{X};\boldsymbol{\lambda}^{(1)}) \\ \vdots \\ \log p(\boldsymbol{X};\boldsymbol{\lambda}^{(M)}) \end{bmatrix}. \tag{4.6}$$

Two mappings that are of particular interest in this thesis are presented next.

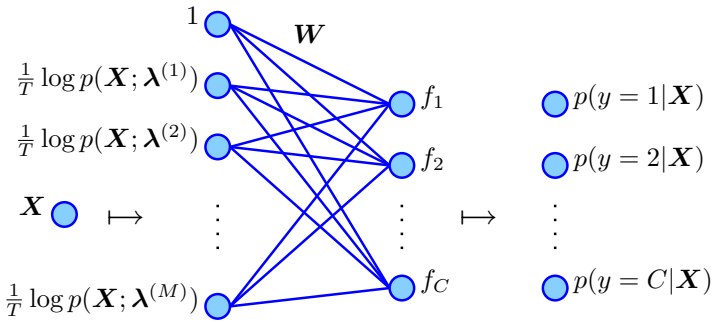

Figure 4.1: The logistic regression model with the likelihood mapping.

The likelihood mapping

We make some minor modifications to the mapping in (4.6) and present the *likelihood mapping* [Birkenes et al., 2006a]

$$\boldsymbol{\phi}_{\mathrm{L}}(\boldsymbol{X};\boldsymbol{\lambda}) = \begin{bmatrix} 1 \\ \frac{1}{T}\log \hat{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(1)}) \\ \vdots \\ \frac{1}{T}\log \hat{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(M)}) \end{bmatrix}, \tag{4.7}$$

where T is the frame length (i.e., the number of vectors in the sequence $\boldsymbol{X}$), and $\hat{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(m)})$ is the Viterbi-approximated likelihood (i.e., the likelihood computed along the Viterbi path) of the mth HMM with parameter vector $\boldsymbol{\lambda}^{(m)}$.

To be more specific, let $\boldsymbol{\lambda}^{(m)} = (\boldsymbol{\pi}^{(m)}, \boldsymbol{A}^{(m)}, \boldsymbol{\eta}^{(m)})$ be the set of parameters for the mth HMM. Then

$$\hat{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(m)}) = \max_{\boldsymbol{q}} p(\boldsymbol{X},\boldsymbol{q};\boldsymbol{\lambda}^{(m)}) \tag{4.8}$$

$$= \max_{\boldsymbol{q}} \pi^{(m)}_{q_1} \prod_{t=2}^{T} a^{(m)}_{q_{t-1},q_t} \prod_{t=1}^{T} p(\boldsymbol{x}_t|q_t;\boldsymbol{\eta}^{(m)}_{q_t}). \tag{4.9}$$

Furthermore, we let each state-conditional probability density function be a Gaussian mixture model (GMM) with a diagonal covariance matrix, i.e.,

$$p(\boldsymbol{x}_t|q_t;\boldsymbol{\eta}^{(m)}_{q_t}) = \sum_{h=1}^{H} c^{(m)}_{qh} \mathcal{N}(\boldsymbol{\mu}^{(m)}_{qh}, \boldsymbol{\Sigma}^{(m)}_{qh}) \tag{4.10}$$

$$= \sum_{h=1}^{H} c^{(m)}_{qh} (2\pi)^{-D/2} \left(\prod_{d=1}^{D} \sigma^{(m)}_{qhd} \right)^{-1} e^{-\frac{1}{2}\sum_{d=1}^{D} \left(\frac{x_{td} - \mu^{(m)}_{qhd}}{\sigma^{(m)}_{qhd}} \right)^2}. \tag{4.11}$$

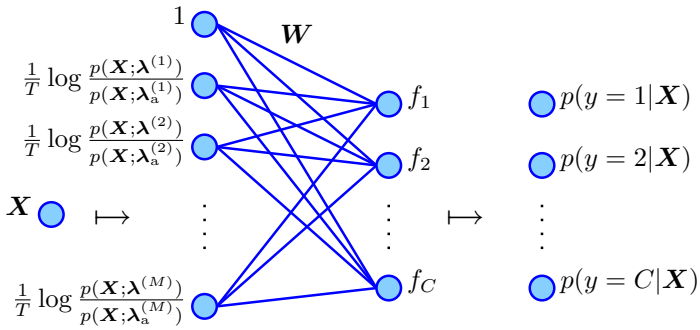

Figure 4.2: The logistic regression model with the likelihood-ratio mapping.

The hyperparameter vector of the likelihood mapping consists of all the parameters of all the HMMs, i.e., $\boldsymbol{\lambda} = (\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(M)})$.

We have chosen to normalize the log-likelihood values with respect to the length T of the sequence $\boldsymbol{X}$. The elements of the likelihood mapping are thus the average log-likelihood per frame for each model. The reason for performing this normalization is that we want utterances of the same word spoken at different speaking rates to map into the same region of space. Moreover, the reason that we use the Viterbi-approximated likelihood instead of the true one is to make it easier to compute its derivatives with respect to the various HMM parameters. These derivatives are needed when we allow the parameters to adapt during training of the logistic regression model.

Figure 4.1 illustrates the logistic regression model with the likelihood mapping. For an observation $\boldsymbol{X}$, the model converts frame-normalized log-likelihoods into posterior probabilities for each class.

Although the likelihood mapping was defined with exactly one HMM per class, this is not a restriction. We may have M models and C classes, where $M \neq C$. Some examples of this will be given in the next chapter.

The likelihood-ratio mapping

The second new mapping we present in this thesis is the *likelihood-ratio mapping* defined as

$$\boldsymbol{\phi}_{\mathrm{LR}}(\boldsymbol{X};\boldsymbol{\lambda}) = \begin{bmatrix} 1 \\ \frac{1}{T}\log\frac{\hat{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(1)})}{\hat{p}(\boldsymbol{X};\boldsymbol{\lambda}_{\mathrm{a}}^{(1)})} \\ \vdots \\ \frac{1}{T}\log\frac{\hat{p}(\boldsymbol{X};\boldsymbol{\lambda}^{(M)})}{\hat{p}(\boldsymbol{X};\boldsymbol{\lambda}_{\mathrm{a}}^{(M)})} \end{bmatrix}, \tag{4.12}$$

where $\hat{p}(x;\boldsymbol{\lambda}^{(m)})$ and $\hat{p}(x;\boldsymbol{\lambda}_{\mathrm{a}}^{(m)})$ denote the Viterbi-approximated likelihood of the model and the *anti-model*, respectively, for the mth class. The anti-model for a class should return high likelihood for observations that are unlikely with respect to the class and low likelihood otherwise. It can be trained, for example, by using all the training data except for the data that are labeled with that particular class. The hyperparameter vector for the likelihood-ratio mapping consists of all the HMM parameters of both models and anti-models, i.e., $\boldsymbol{\lambda} = (\boldsymbol{\lambda}^{(1)},\ldots,\boldsymbol{\lambda}^{(M)},\boldsymbol{\lambda}_{\mathrm{a}}^{(1)},\ldots,\boldsymbol{\lambda}_{\mathrm{a}}^{(M)})$.

Figure 4.2 illustrates the logistic regression model with the likelihood-ratio mapping. In this model, a vector of frame-normalized log-likelihood ratios are converted into posterior probabilities for each word.

In comparison with the likelihood-mapping, the likelihood-ratio mapping involves twice as many models, and has therefore about twice the computational complexity. On the other hand, the likelihood-ratio is expected to be more robust to variations in acoustic conditions than the likelihood [Smith and Gales, 2002].

As with the likelihood mapping, we are not restricted to have the same number of models as the number of classes. In the next chapter, we present experimental results on a phone recognition task where we use $M = 15$ likelihood-ratio detectors as regressors and $C = 39$ phonetic classes.

4.1.1 Adaptive Regressor Parameters

As we saw in Chapter 2, additional discriminative power can be achieved by updating the regressor parameters $\boldsymbol{\lambda}$ in addition to the weight matrix $\boldsymbol{W}$. To this end we need the gradient of the criterion function $\mathcal{P}_\delta^{\log}(\boldsymbol{W},\boldsymbol{\lambda};\mathcal{D})$ in (2.63) with respect to $\boldsymbol{\lambda}$. We saw in Lemma 2.1.4 that the partial derivative of $\mathcal{P}_\delta^{\log}(\boldsymbol{W},\boldsymbol{\lambda};\mathcal{D})$ with respect to λ_l satisfies

$$\frac{\partial}{\partial\lambda_l}\mathcal{P}_\delta^{\log}(\boldsymbol{W};\boldsymbol{\lambda};\mathcal{D}) = \operatorname{trace}\left\{\Big(\frac{\delta}{N}\boldsymbol{\Phi}^T\boldsymbol{W}\boldsymbol{\Gamma} + \boldsymbol{P}^{\mathrm{T}}(\boldsymbol{W}) - \boldsymbol{Y}^{\mathrm{T}}\Big)\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial\lambda_l}\boldsymbol{\Phi}\right\}, \tag{4.13}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} | & & | \\ \boldsymbol{\phi}(\boldsymbol{X}^{(1)};\boldsymbol{\lambda}) & \cdots & \boldsymbol{\phi}(\boldsymbol{X}^{(N)};\boldsymbol{\lambda}) \\ | & & | \end{bmatrix}. \tag{4.14}$$

Thus, what remains in order to compute the partial derivative in (4.13) is the expression for the partial derivatives of the elements of the vectors $\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})$ with respect to each parameter λ_l.

For the likelihood mapping and the likelihood-ratio mapping, the regressor parameters are the HMM parameters, which have certain constraints on which values they can take on. For example, all variances must be greater

than zero. In order to use unconstrained optimization methods such as the steepest descent method or the RProp method [Riedmiller and Braun, 1993], we choose to first transform the parameters to a space where any value is valid. Then we can apply the unconstrained optimization method with these transformed parameters, and finally transform the parameters back to the original space. We use the same transformations as in [Juang et al., 1997], e.g., $\sigma \mapsto \tilde{\sigma} = \log \sigma$ and $\mu \mapsto \tilde{\mu} = \mu/\sigma$ for variances and means, respectively. Thus, after choosing an initial parameter vector $\boldsymbol{\lambda}^0$, we use the mapping $\boldsymbol{\lambda}^0 \mapsto \tilde{\boldsymbol{\lambda}}^0$ to transform the parameters to a space where any value is valid and iterate according to

$$\tilde{\lambda}_l^i = \tilde{\lambda}_l^{i-1} + \Delta\tilde{\lambda}_l^{i-1} \quad \text{for } l = 1, \ldots, L, \tag{4.15}$$

where the step $\Delta\tilde{\lambda}_l^{i-1}$ is defined in either the steepest descent method or the RProp method. Finally, the resulting vector $\tilde{\boldsymbol{\lambda}}^*$ is mapped back to the original parameter space, i.e., $\tilde{\boldsymbol{\lambda}}^* \mapsto \boldsymbol{\lambda}^*$.

For both the likelihood mapping and the likelihood-ratio mapping, we need the partial derivative of the Viterbi-approximated log-likelihood with respect to each transformed HMM parameter. This is given in the following two lemmas for the means and the standard deviations, respectively.

Lemma 4.1.1 *The partial derivative of the Viterbi-approximated log-likelihood with respect to a mean value is*

$$\frac{\partial \log \hat{p}(\boldsymbol{X}^{(n)}; \boldsymbol{\lambda}_m)}{\partial \tilde{\mu}_{qhd}^{(m)}} = \sum_{t=1}^{T} \delta(q - \hat{q}_t) \frac{c_{qh}^{(m)} \mathcal{N}(\boldsymbol{\mu}_{qh}^{(m)}, \boldsymbol{\Sigma}_{qh}^{(m)})}{p(\boldsymbol{x}_t^{(n)} | q, \boldsymbol{\eta}_q^{(m)})} \cdot \frac{(x_{td}^{(n)} - \mu_{qhd}^{(m)})}{\sigma_{qhd}^{(m)}}. \tag{4.16}$$

Proof See App. A

Lemma 4.1.2 *The partial derivative of the Viterbi-approximated log-likelihood with respect to a standard deviation value is*

$$\frac{\partial \log \hat{p}(\boldsymbol{X}^{(n)}; \boldsymbol{\lambda}_m)}{\partial \tilde{\sigma}_{qhd}^{(m)}} = \sum_{t=1}^{T} \delta(q - \hat{q}_t) \frac{c_{qh}^{(m)} \mathcal{N}(\boldsymbol{\mu}_{qh}^{(m)}, \boldsymbol{\Sigma}_{qh}^{(m)})}{p(\boldsymbol{x}_t^{(n)} | q, \boldsymbol{\eta}_q^{(m)})} \cdot \left(\left(\frac{x_{td}^{(n)} - \mu_{qhd}^{(m)}}{\sigma_{qhd}^{(m)}} \right)^2 - 1 \right). \tag{4.17}$$

Proof See App. A

4.2 Kernel Logistic Regression

In kernel logistic regression the mapping $\phi : \mathcal{X} \to \mathbb{R}^{M+1}$ is defined through a symmetric and positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Here, we are concerned with the application of kernel logistic regression to automatic speech recognition, so we let $\mathcal{X}$ be the space of all time series extracted from speech signals. The kernel is thus a function that maps a pair of time series into a real number.

We will in the following consider two approaches for designing a kernel for speech signals. In the first approach, the mapping ϕ is defined explicitly, and a *vector kernel* is chosen that operates on two mapped vectors $\boldsymbol{\phi}(\boldsymbol{X})$ and $\boldsymbol{\phi}(\boldsymbol{X}')$. We use the term *vector kernel* to mean a kernel that operates on two fixed-dimensional vectors, that is, $k : \mathbb{R}^{M+1} \times \mathbb{R}^{M+1} \mapsto \mathbb{R}$. Standard vector kernels such as linear, polynomial and Gaussian kernels may be used.

In the second approach, the kernel is designed from a similarity measure between two time series. We will call such a kernel a *sequence kernel* since it directly operates on two time series which are sequences of vectors. Note that in this case, the mapping $\phi : \mathcal{X} \to \mathbb{R}^{M+1}$ is implicitly defined through the kernel. There is no need to know what this mapping is, only its existence, and that is guaranteed by Mercer's Theorem.

4.2.1 Vector Kernels

In the previous section, we saw some examples of how we can map a time series $\boldsymbol{X}$ into a fixed dimension vector $\boldsymbol{\phi}(\boldsymbol{X})$. Once such a mapping is defined, training and prediction using PLR are straightforward. Alternatively, as we saw in Chapter 2, with ϕ and $\boldsymbol{\Sigma}$ defined, the same result can be obtained using KLR with the kernel

$$k(\boldsymbol{X}, \boldsymbol{X}') = \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X})\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}(\boldsymbol{X}'). \tag{4.18}$$

Perhaps the most well-known example of such a kernel is the Fisher kernel ([Jaakkola and Haussler, 1999a]), where the mapping from a time series $\boldsymbol{X}$ to a vector is $\boldsymbol{\phi}_{\mathrm{Fisher}}(\boldsymbol{X}) = \nabla_{\boldsymbol{\lambda}} \log P(\boldsymbol{X}; \boldsymbol{\lambda})$ as given in (4.3), and $\boldsymbol{\Sigma} = E\{\boldsymbol{\phi}(\boldsymbol{X})\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X})\}$ is the Fisher information matrix. The Fisher information matrix can be approximated by the sample moment matrix given in (2.18), or it can be omitted altogether for which case the Fisher kernel is simply the inner product $k_{\mathrm{Fisher}}(\boldsymbol{X}, \boldsymbol{X}') = \nabla_{\boldsymbol{\lambda}} \log P(\boldsymbol{X}; \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}}^{\mathrm{T}} \log P(\boldsymbol{X}'; \boldsymbol{\lambda})$.

The use of a kernel on the inner product form in (4.18) with an explicitly defined mapping ϕ amounts to finding a linear separating hyperplane in the transformed observation space. Higher order decision boundaries can be obtained by using a non-linear vector kernel on the vectors $\boldsymbol{\psi}(\boldsymbol{X}) =$

$\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})$. For example, using the polynomial kernel in (2.98) yields a new kernel

$$\tilde{k}(\boldsymbol{X},\boldsymbol{X}') = (\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X})\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}(\boldsymbol{X}')+1)^d \tag{4.19}$$

$$= (k(\boldsymbol{X},\boldsymbol{X}')+1)^d, \tag{4.20}$$

that can be used in the KLR model.

4.2.2 Sequence Kernels

Another approach to design a kernel for speech signals is to construct a similarity measure between two speech signals. In the following, we will present kernels that compute a similarity between pairs of time series extracted from speech signals. In this way of specifying a kernel, there is no need to explicitly define the mapping ϕ. We only need to know that such a mapping exists, and that is guaranteed as long as the kernel is symmetric and positive definite. A kernel that operates on two sequences is called a *sequence kernel* and can be directly used in KLR or any other kernel method such as the support vector machine (SVM). The sequence kernels we will present in the following are all motivated by the *dynamic time warping* (DTW) distortion measure between two time series that has been extensively studied for speech applications [Rabiner and Juang, 1993]. We will therefore start by giving a brief review of DTW before we proceed to the various kernels.

An important concept in DTW is the notion of an *alignment* between two time series. The alignment π of length p between two time series $\boldsymbol{X} = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T)$ and $\boldsymbol{X}' = (\boldsymbol{x}'_1,\ldots,\boldsymbol{x}'_{T'})$ is defined as a sequence of tuples $\pi = \big(\pi(1),\ldots,\pi(p)\big)$ with $\pi(i) = \big(\pi_{\boldsymbol{X}}(i),\pi_{\boldsymbol{X}'}(i)\big) \in \{1,\ldots,T\}\times\{1,\ldots,T'\}$. Given such an alignment π and a local distance measure d, e.g., $d(\boldsymbol{x}_i,\boldsymbol{x}'_j) = \|\boldsymbol{x}_i - \boldsymbol{x}'_j\|^2$, the *alignment distance* along π is defined as

$$D_\pi(\boldsymbol{X},\boldsymbol{X}') = \frac{1}{M_\pi}\sum_{i=1}^{p} m(i)d(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)},\boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}), \tag{4.21}$$

where $m(i)$ is a non-negative path weight, and M_π is a path normalization factor. The path normalization factor is usually taken to be $M_\pi = \sum_{i=1}^{p} m(i)$, and $m(i)$ is chosen such that M_π is independent of π for reasons that will become clear later. The *DTW distance* $D(\boldsymbol{X},\boldsymbol{X}')$ is defined as the minimum alignment distance over a set $\mathcal{A}(\boldsymbol{X},\boldsymbol{X}')$ of possible

alignments, that is,

$$D(\boldsymbol{X}, \boldsymbol{X}') = \min_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} D_\pi(\boldsymbol{X}, \boldsymbol{X}') \tag{4.22}$$

$$= \min_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) d(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}). \tag{4.23}$$

We will here restrict ourselves to the set $\mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')$ of alignments satisfying

$$1 = \pi_{\boldsymbol{X}}(1) \leq \cdots \leq \pi_{\boldsymbol{X}}(p) = T \tag{4.24}$$

$$1 = \pi_{\boldsymbol{X}'}(1) \leq \cdots \leq \pi_{\boldsymbol{X}'}(p) = T', \tag{4.25}$$

with unitary increments and no simultaneous repetitions. This means that each alignment starts in $\pi(1) = (1, 1)$ and ends in $\pi(p) = (T, T')$, with each step satisfying

$$\pi(i) - \pi(i-1) \in \{(0, 1), (1, 0), (1, 1)\}. \tag{4.26}$$

Such alignments are known as *global alignments*, as opposed to *local alignments* that can start and end at arbitrary pairs of indexes as long as each step satisfies (4.26).

If the path weights $m(i)$ are chosen such that the path normalizing factor M_π is independent of π, we can move $1/M_\pi$ outside the minimization operator in (4.23) and the DTW distance can be efficiently computed using dynamic programming. An example of such a choice is $m(1) = 2$ and

$$m(i) = \begin{cases} 1, & \text{if } \pi(i) - \pi(i-1) \in \{(0, 1), (1, 0)\} \\ 2, & \text{if } \pi(i) - \pi(i-1) = (1, 1) \end{cases} \quad \text{for } i = 2, \ldots, p. \tag{4.27}$$

Then $M_\pi = \sum_{i=1}^{p} m(i) = T + T'$ is independent of the particular alignment, and the dynamic programming algorithm for computing $D(\boldsymbol{X}, \boldsymbol{X}')$ is given in the following algorithm.

Algorithm 4.2.1 (The dynamic programming algorithm for computing the DTW distance) *Computes the DTW distance $D(\boldsymbol{X}, \boldsymbol{X}')$ of two time series $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ and $\boldsymbol{X}' = (\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{T'})$.*

1. *Initialize: for $i = 0, \ldots, T$ and $j = 0, \ldots, T'$:*

$$D_{i,0} = D_{0,j} = 0 \tag{4.28}$$

2. Iterate: for $i = 1, \ldots, T$ and $j = 1, \ldots, T'$:

$$D_{i,j} = \min \left\{ \begin{array}{c} D_{i-1,j} + d(\boldsymbol{x}_i, \boldsymbol{x}'_j), \\ D_{i-1,j-1} + 2d(\boldsymbol{x}_i, \boldsymbol{x}'_j), \\ D_{i,j-1} + d(\boldsymbol{x}_i, \boldsymbol{x}'_j) \end{array} \right\} \tag{4.29}$$

3. Terminate:

$$D(\boldsymbol{X}, \boldsymbol{X}') = \frac{D_{T,T'}}{T + T'} \tag{4.30}$$

The alignment distance along an alignment path π was defined in (4.21) as a weighted sum of local distances computed from a local distance measure d. If we substitute the local distance measure d in (4.21) with a local similarity measure κ, we can construct a similarity measure between two time series in the same fashion as we constructed a distance measure between two time series in the above discussion. Thus, we define the *alignment score* of the alignment π as

$$S_\pi(\boldsymbol{X}, \boldsymbol{X}') = \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}), \tag{4.31}$$

where κ is a *conditionally positive definite kernel* (see the definition in Section 2.2.1).

A natural first choice for a kernel between two time series is then the *dynamic time-alignment kernel* (DTAK) introduced in [Shimodaira et al., 2002]:

$$k_{\mathrm{DTAK}}(\boldsymbol{X}, \boldsymbol{X}') = \max_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} S_\pi(\boldsymbol{X}, \boldsymbol{X}') \tag{4.32}$$

$$= \max_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}). \tag{4.33}$$

Since the only differences from the DTW distance is that a local kernel κ is used instead of a local distance d, and a max operator is used instead of a min operator, the computation of the DTAK kernel can be done using Algorithm 4.2.1 with only minor modifications. It should be made clear however, that the DTAK kernel is not guaranteed to be a positive definite kernel [Vert et al., 2004]. Nevertheless, a modification can be made to the kernel matrix so that it becomes positive definite. We will see later how this can be done.

A similar kernel can be made by maximizing the exponent of the alignment score instead of maximizing the alignment score itself. This is the

same as exponentiating the DTAK kernel, i.e.,

$$k_{\mathrm{DTW}}(\boldsymbol{X}, \boldsymbol{X}') = \max_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \exp\{S_\pi(\boldsymbol{X}, \boldsymbol{X}')\} \tag{4.34}$$

$$= \max_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \exp\Big\{ \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}) \Big\} \tag{4.35}$$

$$= \exp\Big\{ \max_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}) \Big\}. \tag{4.36}$$

Also this kernel lacks guarantees of the positive definiteness property. If we let κ be the conditionally positive definite kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}'_j) = -\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|^2$ and $m(i) = 1$, we get the *Gaussian dynamic time warping (GDTW) kernel* introduced in [Bahlmann et al., 2002].

In both kernels presented up to this point, the similarity measure between two time series is found by maximizing the alignment score $S_\pi(\boldsymbol{X}, \boldsymbol{X}')$ among all alignment paths $\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')$. One might argue that a better similarity measure is one that adds up the scores of all alignment paths. In this sense, not only the largest score contributes to the overall similarity measure, but the scores of all alignment paths. A kernel that has this property is

$$k_{\mathrm{DTW}^*}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \exp\{S_\pi(\boldsymbol{X}, \boldsymbol{X}')\} \tag{4.37}$$

$$= \sum_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \exp\Big\{ \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}) \Big\} \tag{4.38}$$

$$= \exp\Big\{ \max\nolimits^*_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}) \Big\}, \tag{4.39}$$

where, given positive scalars $z_1, \ldots, z_L$, we define $\max^* z_i = \log \sum \exp z_i$. This kernel can be shown to be positive definite under mild conditions on the conditional positive definite kernel κ for specific choices of the path weights $m(i)$ and the path normalization factor M_π. For example, if we set $m(i) = 1$ and $M_\pi = 1$ we obtain the *global alignment (GA) kernel*

introduced in [Cuturi et al., 2007]:

$$k_{\mathrm{GA}}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \exp\Big\{ \sum_{i=1}^{p} \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}) \Big\} \tag{4.40}$$

$$= \sum_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \prod_{i=1}^{p} \exp\Big\{ \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}) \Big\} \tag{4.41}$$

$$= \sum_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')} \prod_{i=1}^{p} k(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}), \tag{4.42}$$

where we have written $k = \exp \kappa$. From Lemma 2.2.4 we know that the kernel k is positive definite since κ is conditionally positive definite. The following theorem states that the GA kernel k_{GA} is positive definite under mild conditions on the kernel k.

Theorem 4.2.2 *Let k be a positive definite kernel such that $\frac{k}{1+k}$ is positive definite. Then k_{GA} is positive definite.*

Proof See [Cuturi et al., 2007].

Remark A positive definite kernel k such that $\frac{k}{1+k}$ is positive definite can be trivially computed by starting with a positive definite kernel $\tilde{k}$ with $|\tilde{k}| < 1$, and defining $k = \sum_{i=1}^{\infty} \tilde{k}^i = \frac{\tilde{k}}{1-\tilde{k}}$. Then k is positive definite and $\frac{k}{1+k} = \tilde{k}$ is positive definite by assumption. An example of such a kernel is

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{\frac{1}{2} \exp\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{\sigma^2}\}}{1 - \frac{1}{2} \exp\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{\sigma^2}\}}, \tag{4.43}$$

where $\tilde{k} = \frac{1}{2} \exp\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{\sigma^2}\}$ is half the Gaussian kernel. In practice, most popular kernels, including the Gaussian kernel and the exponential of the Gaussian kernel, have the property that $\frac{k}{1+k}$ yields a positive definite kernel matrix, which in an experimental context will be sufficient [Cuturi et al., 2007].

The following dynamic programming algorithm can be used to compute the value of the global alignment kernel for a pair of time series. The computational efficiency is nearly as good as Algorithm 4.2.1.

Algorithm 4.2.3 (The dynamic programming algorithm for computing the global alignment kernel) *Computes the global alignment kernel $k_{\mathrm{GA}}(\boldsymbol{X}, \boldsymbol{X}')$ of two time series $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ and $\boldsymbol{X}' = (\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{T'})$.*

1. *Initialize: for* $i = 0, \ldots, T$ *and* $j = 0, \ldots, T'$*:*

$$S_{i,0} = S_{0,j} = 0 \tag{4.44}$$

2. *Iterate: for* $i = 1, \ldots, T$ *and* $j = 1, \ldots, T'$*:*

$$S_{i,j} = (S_{i-1,j} + S_{i-1,j-1} + S_{i,j-1})k(\boldsymbol{x}_i, \boldsymbol{x}'_j) \tag{4.45}$$

3. *Terminate:*

$$k_{\mathrm{GA}}(\boldsymbol{X}, \boldsymbol{X}') = S_{T,T'} \tag{4.46}$$

In many cases of practical interest, the kernel in (4.39) suffers from the diagonal dominance issue, meaning that $k_{\mathrm{DTW}^*}(\boldsymbol{X}, \boldsymbol{X})$ is orders of magnitude larger than $k_{\mathrm{DTW}^*}(\boldsymbol{X}, \boldsymbol{X}')$ for two different sequences $\boldsymbol{X}$ and $\boldsymbol{X}'$. It has been observed in practice that many kernel methods perform poorly in this situation [Vert et al., 2004]. A possible solution is take the logarithm of the kernel in (4.39), giving

$$\log k_{\mathrm{DTW}^*}(\boldsymbol{X}, \boldsymbol{X}') = \max_{\pi \in \mathcal{A}(\boldsymbol{X}, \boldsymbol{X}')}{}^{*} \frac{1}{M_\pi} \sum_{i=1}^{p} m(i) \kappa(\boldsymbol{x}_{\pi_{\boldsymbol{X}}(i)}, \boldsymbol{x}'_{\pi_{\boldsymbol{X}'}(i)}). \tag{4.47}$$

By doing this however, the kernel is no longer positive definite, even for $m(i) = 1$ and $M_\pi = 1$. Nevertheless, the kernel matrix can be made positive definite by adding to the diagonal the magnitude of the most negative eigenvalue. The new kernel matrix will then have only non-negative eigenvalues, meaning that it is positive definite. The same trick can also be applied to the other sequence kernels presented in this section that are not positive definite.

4.3 Experiments

Experiments on two different speech recognition tasks were performed; 1) recognition of the English letters {B,C,D,E,G,P,T,V,Z}, known as the *E-set*, spoken in isolation, and 2) recognition of phones with known segment boundaries, also known as phone classification. For the former set of experiments, the E-set of the TI46 database was used. A thorough analysis of the various approaches to isolated-word speech recognition using logistic regression was performed. The latter experiment on phone classification was performed using the TIMIT database [Lamel et al., 1986].

4.3.1 A Thorough Analysis on the TI46 E-set

In this section we give a thorough analysis on the use of logistic regression for recognition of the letters in the TI46 E-set spoken in isolation. We start by giving a description of the TI46 database before we explain our baseline HMM system. Then, we present results from experiments using PLR with fixed regressor parameters followed by PLR with adaptive regressor parameters. Finally, we present results using KLR with vector kernels and sequence kernels.

The TI46 database and the baseline system

The E-set of the TI46 database consists of utterances from 16 different speakers (8 male and 8 female) of the letters in the highly confusable E-set {B,C,D,E,G,P,T,V,Z} spoken in isolation. There are 26 utterances of each letter from each speaker, of which 10 utterances are designated for training and 16 for test. This would give a total of $16 * 9 * 10 = 1440$ utterances for training and $16 * 9 * 16 = 2304$ for test. In the experiments presented here however, some utterances were missing, so 1433 utterances were used for training and 2291 were used for test.

From each speech signal, a sequence of feature vectors were extracted using a 25 ms Hamming window and a window shift of 10 ms. Each feature vector consisted of 13 Mel-frequency cepstral coefficients (MFCC), including the 0th cepstral coefficient, augmented with their delta and acceleration coefficients. This resulted in 39-dimensional vectors.

As a baseline system, we estimated one HMM for each of the 9 letters in the E-set. Each model was a left-to-right HMM with 6 states, and with a Gaussian mixture model (GMM) with 5 mixtures in each state having diagonal covariance matrices. The models were trained using the maximum likelihood (ML) criterion. Using the standard generative classification approach to do recognition on the test set, that is, choosing the letter whose model has the highest likelihood for each test utterance, an accuracy of 88.3% was obtained. Equal prior probabilities for the letters were used.

Penalized logistic regression with fixed regressor parameters

A series of experiments using PLR with fixed regressor parameters were conducted. In the first experiment, the likelihood mapping was compared with the likelihood-ratio mapping. The 9 baseline HMMs, one for each letter in the E-set, were used in the likelihood mapping. The likelihood-ratio mapping used in addition one antimodel for each of the 9 letters. Each antimodel was trained using maximum likelihood estimation by leaving out only the training data belonging to the particular class of the antimodel

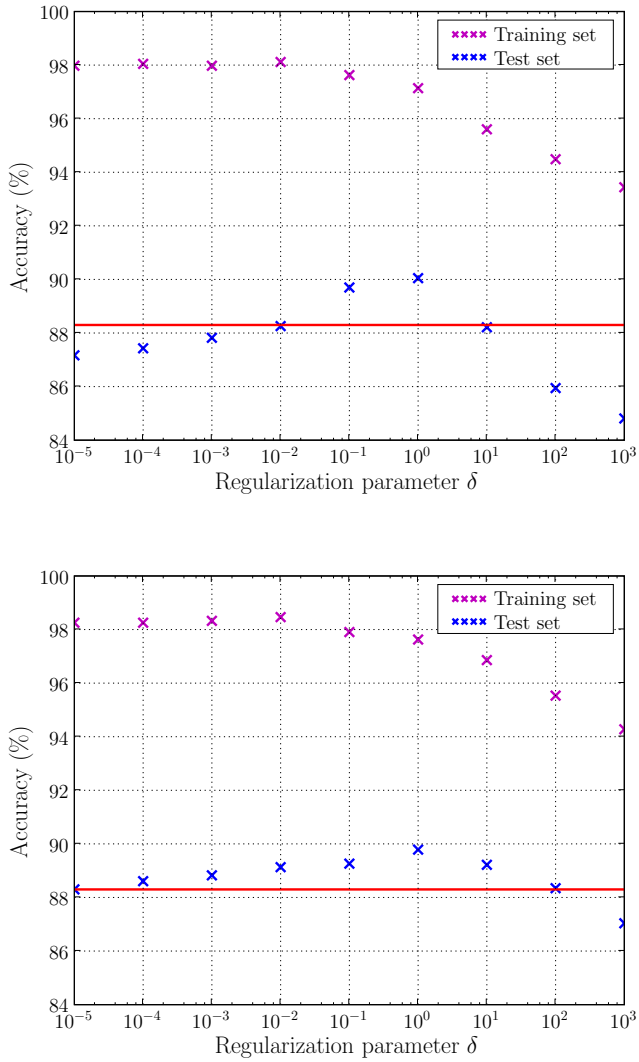

Figure 4.3: Accuracy on the TI46 E-set for various values of δ using the likelihood mapping (top) and likelihood-ratio mapping (bottom) with fixed regressor parameters. The red line represents the baseline accuracy of 88.3%.

being trained. With both mappings, the weight matrix $\boldsymbol{W}$ of the logistic regression model was trained using the penalized logistic regression machine (PLRM). We used 50 Newton iterations with an adaptive stepsize α_i, and 500 iterations in the conjugate gradient (CG) method. At each Newton step, the stepsize was initialized to $\alpha_i = 1/\|\boldsymbol{K}\|$, where $\boldsymbol{K}$ is the Gram matrix, and divided by 2 until a decrease in the criterion function occurred. Figure 4.3 shows the accuracy of PLR with the likelihood mapping (top) and the likelihood-ratio mapping (bottom) for various values of the regularization parameter δ. It can be seen that the best accuracy for the likelihood mapping on the test set is 90.0%, while it is 89.8% for the likelihood-ratio mapping. Both approaches achieve maximum accuracy for $\delta = 1.0$. Although the best accuracy of the likelihood-ratio mapping is not as good as the best accuracy of the likelihood mapping, the likelihood-ratio mapping demonstrates better robustness against δ. Nevertheless, when also taking into account the computational complexity, we chose to perform the rest of the experiments presented in this chapter using the likelihood mapping.

Let us consider the results obtained with the likelihood mapping in more detail. As δ decreases below 1.0, the training accuracy increases and converges to about 98.0%, while the test accuracy decreases and converges to about 87% accuracy. These are the accuracies resulting from the maximum likelihood estimate (without penalty) of the weight matrix $\boldsymbol{W}$, since in this case the penalty term in the criterion function goes to zero. For such small values of δ, the low accuracy on the test set are caused by overfitting to the training data. In the other end, increasing δ beyond 1.0 results in a dramatic decrease in both the training accuracy and the test accuracy for both mappings. This happens because the penalty term increasingly dominates over the criterion function causing the training data to be less important for the optimization procedure.

An important issue in PLR is to determine the value of the regularization parameter δ which results in the highest accuracy on the test set. Only the training set may be used for this, while the test set is assumed to be unseen during training and development. We compared two approaches: 1) maximization of the average 10-fold cross-validation accuracy on the training set, and 2) minimization of an information criterion (ABIC) presented in Chapter 2. Figure 4.4 shows the average 10-fold cross-validation accuracy (top) and ABIC (bottom) computed on the training set for the same range of delta values as in 4.3. The likelihood mapping was used. It can be seen that the best δ value according to the cross-validation method is $\delta = 0.01$, while it is $\delta = 0.1$ for the ABIC method. Since the best test accuracy occurs for $\delta = 1.0$ (see the top plot of Figure 4.3), the ABIC method performs better in estimating δ than the cross-validation method in this particular experiment.

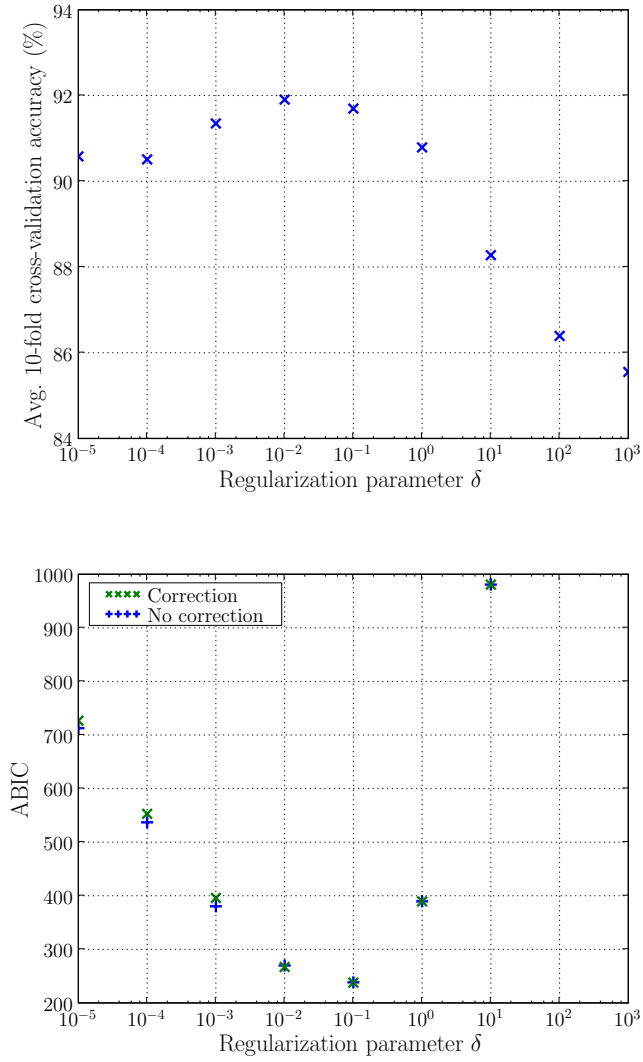

Figure 4.4: Top: Average 10-fold cross-validation accuracy on the training data for various values of δ. The maximum occurs for $\delta = 0.01$. Bottom: The ABIC criterion for various values of δ. The minimum occurs for $\delta = 0.1$.

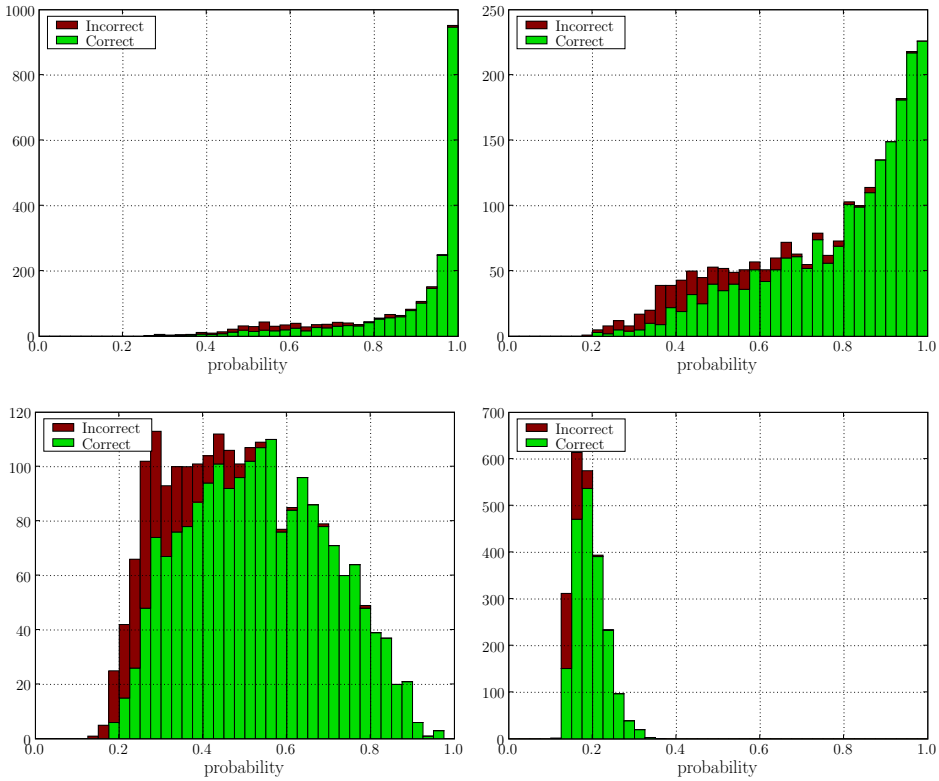

Figure 4.5: Histogram of the highest posterior probabilities for all of the test utterances for $\delta = 1.0$ (top left), $\delta = 10.0$ (top right), $\delta = 100.0$ (bottom left), and $\delta = 1000.0$ (bottom right). All histograms were generated using the likelihood mapping with fixed regressor parameters.

We have plotted ABIC with and without correction term. The correction term was approximated using importance sampling with $J = 1000000$ samples. From the figure, it can be seen that the correction term is negligible and does not influence the minimization of the criterion. This suggests that the quadratic approximation of the criterion function may be sufficient.

A major advantage of logistic regression compared to many other classification methods is that we obtain conditional probabilities of each class given an observation. Figure 4.5 shows four histograms, one for each $\delta \in \{1, 10, 100, 1000\}$, of the highest conditional probabilities (i.e., the probabilities of the recognition decisions) for all of the test utterances. Each bar in the histograms shows the number of utterances resulting in a recognition decision probability within an interval defined by the left and right edges of the bar. Each bar has two colors, light green and dark red, where green

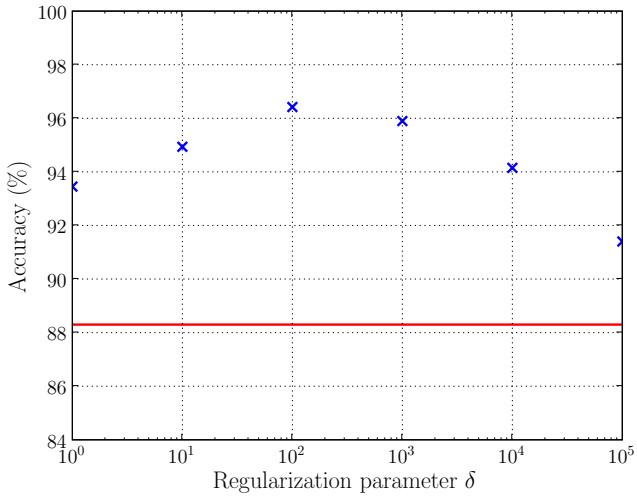

Figure 4.6: Accuracy on the test set of the TI46 E-set after 1000 coordinate descent (CD) iterations. The steepest descent method with 1 iteration was used to update the HMM means in every CD iteration.

represents the portion of these probabilities which resulted in a correct decision, and red represents the portion of probabilities which resulted in an incorrect decision. The main observation from studying these plots is that the effective dynamic range of the probabilities is strongly dependent on δ. For small values of δ, the highest conditional probability for each observation is close to 1, which forces the other classes to have probabilities close to 0. Large values of δ result in probabilities that are close to $1/C$, i.e., around 0.11 for the E-set since there are $C = 9$ classes. It can also be seen from Figure 4.5 that probabilities resulting in an incorrect decision are generally lower than probabilities resulting in a correct decision. This suggests that the probability of the decision may be used as a confidence measure or for verification purposes.

Penalized logistic regression with adaptive regressor parameters

Next we did experiments with adaptive regressor parameters. Again the likelihood mapping was used and the associated HMMs were initially trained using the maximum likelihood criterion. Only the mean values of the HMMs were adapted, while the other parameters remained fixed. First, the weight matrix $\boldsymbol{W}_0$ was estimated using PLRM with 10 Newton iterations each having 200 CG iterations. Then, a coordinate descent (CD) approach was used, where each (CD) iteration consisted of first a step in

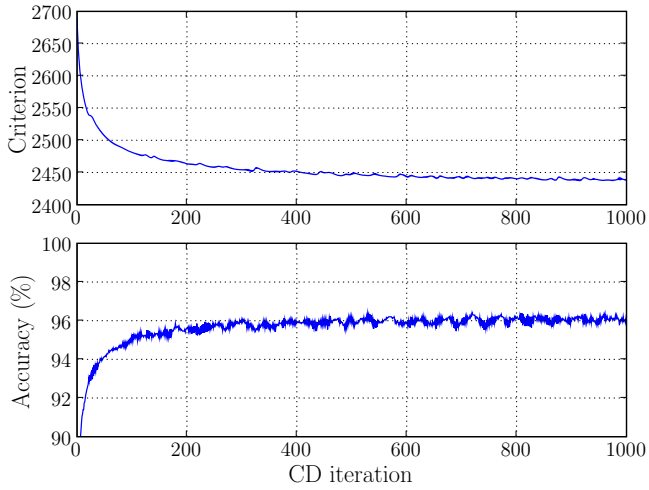

Figure 4.7: Criterion (top) and accuracy on the test set (bottom) of the TI46 E-set with $\delta = 1000$. The steepest descent method with 1 iteration was used to update the HMM means in every CD iteration.

$\boldsymbol{\lambda}$ space and then a step in $\boldsymbol{W}$ space. As a preliminary experiment, we decided to try the steepest descent (SD) method for the minimization in the $\boldsymbol{\lambda}$ space with one iteration and stepsize 0.1. The successive steps in the $\boldsymbol{W}$ space were computed using PLRM with three Newton iterations each having 200 CG iterations, with the initial weight matrix being the weight matrix from the last coordinate descent iteration. Figure 4.6 shows the accuracy on the test set after 1000 coordinate descent iterations for $\delta \in \{1, 10, 100, 1000, 10000, 100000\}$. The best accuracy obtained is 96.4% which was achieved for $\delta = 100$.

The above results were compared with the results obtained with the minimum classification error (MCE) approach in [Juang et al., 1997]. The best accuracy achieved with MCE was found to be 95.0% and was achieved with the parameters (using the notation in [Juang et al., 1997]) $\eta = 100$, $\gamma = 150$ and $\theta = 6$. The steepest descent method with a fixed stepsize and 500 iterations was used for the optimization. Table 4.1 summarizes the best accuracy obtained with the PLR approach described above with $\delta = 100$ and after 1000 iterations, along with the baseline and the MCE approach. Note that the PLR parameter δ and the MCE parameters η, γ and θ were adjusted so as to give the highest test accuracy. The table shows that the combined generative-discriminative approaches (the MCE and the PLR approach) outperformed the generative baseline approach

considerably. Moreover, the PLR approach achieved better accuracy than the MCE approach.

Table 4.1: Best accuracy on the test set of the TI46 E-set

Baseline	MCE	PLR with adaptive regressors (SD)
88.3%	95.0%	96.4%

Let us now examine how the criterion function for PLR and the test accuracy vary as functions of the number of CD iterations. These are shown in Figure 4.7 for $\delta = 1000$, where the evolution of the criterion function is at the top and the evolution of the test accuracy is at the bottom. The accuracy on the training set converged to 100% after about 100 iterations and is not shown. It can be seen from the figure that the accuracy on the test set increased relatively fast the first few iterations, and then increased more slowly as the number of iterations got larger. After 1000 iterations an accuracy of 95.9% was reached. The erratic behaviour of the accuracy in Figure 4.7 is believed to be caused by the fixed stepsize of 0.1 in the steepest descent method. With this, the optimization algorithm never converged to a local minimum, but rather took large steps around it. Of course, with a smaller stepsize this behaviour will be reduced, but then at the expense of slower convergence. Another approach would be to use a larger number of SD iterations for each CD iteration with an adaptive stepsize. We tried several alternative SD approaches, but soon decided to switch to a different optimization algorithm due to slow convergence and the difficulties with the setting of the stepsize.

We decided to substitute the SD method with the RProp algorithm [Riedmiller and Braun, 1993]. We used 100 iterations in the RProp method for each coordinate descent iteration, with the initial stepsize set to 0.01. Figure 4.8 shows the criterion function (top) and the accuracy on the test set (bottom) as functions of the number of coordinate descent iterations for $\delta = 1000$. If we compare these plots with the similar plots for the steepest descent method in Figure 4.7, we can see that the RProp method achieves a lower value of the criterion function than the lowest value obtained with the steepest descent method after only 4 CD iterations, or 400 RProp iterations. Moreover, the criterion function continues to decrease to a value just above 2400 after 100 CD iterations, while the SD method did not get lower in the criterion function than about 2440. However, the accuracy on the test set climbs to 96.7% after the 4th CD iteration, and then radically decreases at the following iterations, ending up at only 90.7% after 100 CD iterations and still decreasing. The decrease in test set accuracy as the number of CD iterations increases is the effect of overfitting to the training data. This effect was not observed with the above SD method, possibly due to the

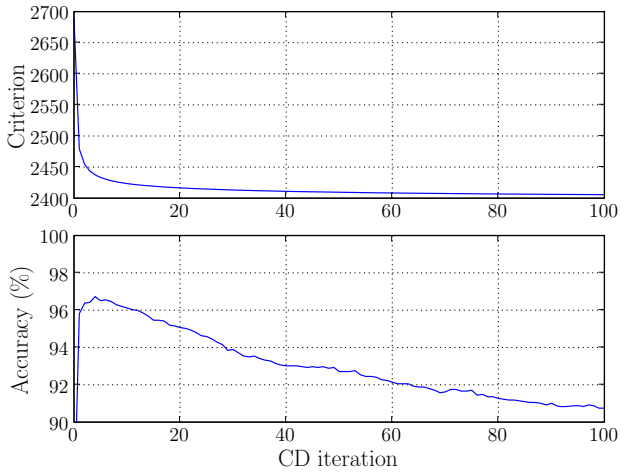

Figure 4.8: Criterion (top) and accuracy on the test set (bottom) of the TI46 E-set using RProp and $\delta = 1000$.

large, fixed stepsize which prevented convergence.

In order to reduce the effect of overfitting to the training data, we attempted the following three approaches: 1) reduce the number of free parameters, 2) train with a penalty term for the regressor parameters, and 3) early stopping. First, we did experiments with reduced number of free parameters. Figure 4.9 shows the criterion function (top) and the accuracy on the test set (bottom) as functions of the number of coordinate descent iterations when HMMs with 2, 3, 4 and 5 mixture components per state were used as regressors. The training accuracy converged to 100% after 70, 4, 3, and 2 CD iterations for 2, 3, 4, and 5 mixtures, respectively. Although reduction in the number of mixture components reduces the effect of overfitting, the problem is still severe. The best accuracy after 100 CD iterations occurred with 2 mixture components and was 93.5%.

Next we considered the use of a penalty term for the regressor parameters. We considered values of the regularization parameter $\delta' \in \{0, 0.1, 1.0\}$, where $\delta' = 0$ implies that there is no penalty on the regressor parameters. The criterion function and the accuracy on the test set as functions of the number of CD iterations are shown in Figure 4.10 for each δ'. The training accuracy converged to 100% after 3 and 2 iterations for $\delta' = 0.1$ and $\delta' = 0$, respectively. For $\delta' = 1.0$, the training accuracy did not converge to 100%, but ended up at 98.8% after 100 CD iterations. Again, the amount of overfitting to the training data has been reduced, but is still severe. It is only $\delta' = 1.0$ that avoids overshooting in the test accuracy, but then at

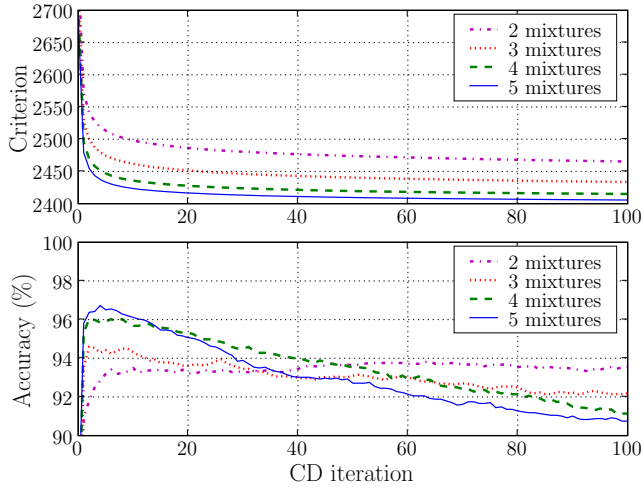

Figure 4.9: Criterion (top) and accuracy on the test set (bottom) of the TI46 E-set using RProp and $\delta = 1000$ using 2, 3, 4, and 5 mixture components in the GMM of each HMM state.

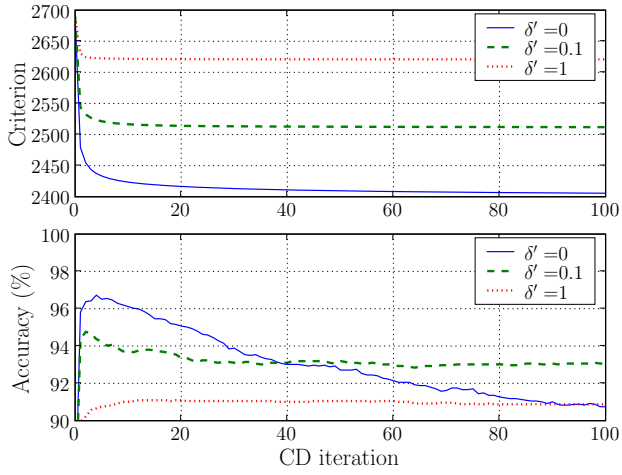

Figure 4.10: Criterion (top) and accuracy on the test set (bottom) of the TI46 E-set using RProp and $\delta = 1000$ using a penalty term for the HMM parameters with parameter $\delta' \in \{0, 0.1, 10\}$.

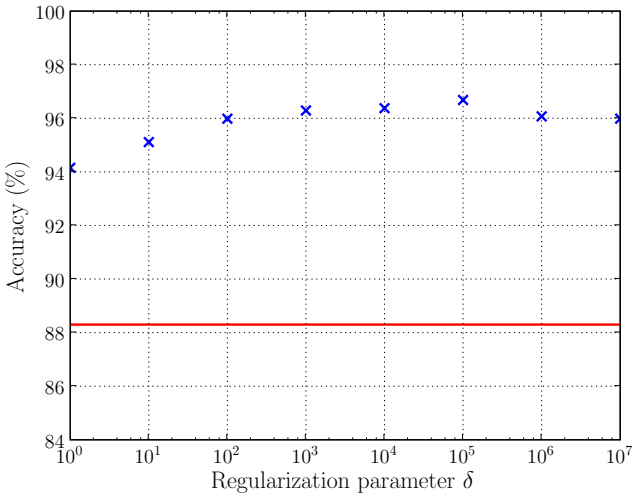

Figure 4.11: Accuracy on the test set of the TI46 E-set using the RProp method with 100 iterations to update the HMM means in every CD iteration. Early stopping was used, where the stop iterations were found using maximum average 10-fold cross validation accuracy on the training set.

the expense of low accuracy. The best accuracy after 100 CD iterations occurred for $\delta' = 0.1$ and was 93.1%.

The last approach we attempted in order to reduce the effect of overfitting to the training data was early stopping. For this, we used 10-fold cross-validation on the training set, and selected the CD iteration number resulting in the largest average cross-validation accuracy. For $\delta = 1000$, the largest average cross-validation accuracy occurred at CD iteration 8 which had a test accuracy of 96.3%. We used the same approach for all the values of δ in the set $\{1, 10, 100, 1000, 10000, 100000, 1000000, 10000000\}$. The results are shown in Figure 4.11. In addition to using the cross-validation method for finding the best stop iteration, it can also be used to determine the best value for δ. Among the δ values in the above set, $\delta = 1000$ resulted in the best average cross validation accuracy.

Kernel logistic regression with vector kernels

In the following we present results on the TI46 E-set using KLR with vector kernels. Two vector kernels were considered; the linear kernel and the Gaussian kernel. Both kernels were defined with the likelihood mapping, with $\boldsymbol{\Sigma}$ set to the identity matrix. Thus, the linear kernel is $k_{\text{lin}}(\boldsymbol{X}, \boldsymbol{X}') = \boldsymbol{\phi}_{\text{L}}(\boldsymbol{X})\boldsymbol{\phi}_{\text{L}}^{\text{T}}(\boldsymbol{X}')$, and the Gaussian kernel is $k_{\text{Gauss}}(\boldsymbol{X}, \boldsymbol{X}') = \exp(\|\boldsymbol{\phi}_{\text{L}}(\boldsymbol{X}) -$

$\boldsymbol{\phi}_{\mathrm{L}}(\boldsymbol{X}')\|^2/(2\sigma^2)$). Training for KLR was done with a nonlinear conjugate gradient (CG) method provided by the authors of [Myrvoll and Matsui, 2006]. We also compared the results with the support vector machine (SVM), which was trained using the LIBSVM library [Chang and Lin, 2001]. Multiclass classification with SVM was done using the one-vs-one approach followed by majority voting. First we considered the linear kernel. The regularization parameters $\delta \in \{10^{-4}, 10^{-3}, \ldots, 10^{2}\}$ for KLR and $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{21}\}$ for SVM, respectively, were optimized so as to obtain the best accuracy on the test set. This happened for $\delta = 0.1$ for KLR and $C = 8$ for SVM. Table 4.2 shows the accuracy on the test set obtained with the linear kernel in KLR and SVM. The baseline result and the best PLR result without updating the regressor parameters are also included for comparison. The table shows that KLR with the linear kernel did not perform as well as PLR. The difference in performance may be explained by the difference in $\boldsymbol{\Sigma}$ for the two approaches, which is the only theoretical difference. In PLR, $\boldsymbol{\Sigma}$ was the sample moment matrix of the transformed observations, while for KLR, $\boldsymbol{\Sigma}$ was chosen to be the identity matrix. There may also be numerical differences between the two optimization algorithms and their implementation. The best result with linear kernel was obtained with SVM.

Table 4.2: Best accuracies on the test set of the TI46 E-set using KLR and SVM with linear kernel compared with PLR and the baseline.

Baseline	PLR	KLR	SVM
88.3%	90.0%	89.1%	91.1%

Next we considered the Gaussian kernel. We let $\gamma = 1/(2\sigma^2)$ in order to conform with the definition of the Gaussian kernel in the software packages. For KLR, the parameters $\gamma \in \{0.1, 1, 10\}$ and $\delta \in \{0.01, 0.1, 1, 10\}$ were optimized so as to obtain the best accuracy on the test set. The best values were $\gamma = 1$ and $\delta = 1$. Similarly for SVM, we chose the best $\gamma \in \{2^{-25}, 2^{-23}, \ldots, 2^{3}\}$ and $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{29}\}$. The best accuracy was obtained for $\gamma = 2^{-19}$ and $C = 2^{21}$. Table 4.3 presents the results obtained using KLR and SVM with the Gaussian kernel compared with the baseline. Surprisingly, KLR with the Gaussian kernel performed considerably worse than the baseline. The reason for this in unclear, and should receive further study. The performance of SVM improved only slightly relative to the case with the linear kernel. This indicates that the transformed observation space is not rich enough for improved performance with the use of nonlinear decision boundaries.

Table 4.3: Best accuracies on the test set of the TI46 E-set using KLR and SVM with the Gaussian kernel compared with the baseline.

Baseline	KLR	SVM
88.3%	83.5%	91.3%

Kernel logistic regression with sequence kernels

We now present experimental results on the TI46 E-set with the use of the GA kernel for KLR, as well as the GA kernel and the DTAK kernel for SVM. Unlike in the experiments presented in the preceding sections, we used here feature vector sequences of only 13-dimensional MFCC, without the inclusion of delta and acceleration coefficients. We present results with the log of the GA kernel as well as the DTAK kernel, both with a Gaussian local kernel $\kappa = \exp(\|\boldsymbol{x} - \boldsymbol{x}'\|^2)/(2\sigma^2)$ defined for two vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$. In all the obtained kernel matrices we needed to subtract from the diagonal the most negative eigenvalue in order to make them positive definite, as this is an assumption of the training procedures. As before, we used the nonlinear CG method provided by the authors in [Myrvoll and Matsui, 2006] for KLR training, and the LIBSVM library [Chang and Lin, 2001] for the SVM training. The kernel parameter $\gamma = 1/(2\sigma^2) \in \{10^{-2}, 15^{-2}, \ldots, 40^{-2}\}$, and the regularization parameters $\delta \in \{0.01, 0.1, 1, 10, 100\}$ and $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{6}\}$, for KLR and SVM, respectively, were optimized so as to obtain the best test accuracy. The optimal values were $\delta = 1$ and $\gamma = 30^{-2}$ for KLR with the GA kernel, $C = 1000$ and $\gamma = 25^{-2}$ for SVM with the GA kernel, and $C = 1000$ and $\gamma = 15^{-2}$ for SVM with the DTAK kernel. Table 4.4 summarizes the results. We make the following observations. Both KLR and SVM with the GA kernel outperform SVM with the DTAK kernel. SVM with the GA kernel outperforms KLR with the GA kernel.

Table 4.4: Best accuracies on the test set of the TI46 E-set using KLR and SVM with the GA kernel as well as SVM with DTAK compared with the baseline.

Baseline	KLR (GA kernel)	SVM (GA kernel)	SVM (DTAK)
88.3%	93.2%	94.6%	88.5%

4.3.2 Phone Classification on the TIMIT Database

In the following we present results on phone classification on the TIMIT database. We used PLR with adaptive regressor parameters, since this approach resulted in the best performance on the TI46 E-set.

The TIMIT database and the baseline system

The TIMIT database [Lamel et al., 1986] consists of 6300 sentences spoken by 630 speakers from 8 major dialect regions of the United States. There are 10 sentences per speaker; 2 dialect sentences (SA), 5 phonetically compact sentences (SX), and 3 phonetically diverse sentences (SI). The database has a predefined portion for training consisting of all the SX and SI sentences from 462 speakers giving a total of 3696 sentences. The sentences from the remaining 168 speakers are meant for development and testing purposes. We will follow [Halberstadt and Glass, 1997] and use the core test set spoken by 24 speakers for testing and the MIT development set spoken by 50 speakers for validation. Only the SX and SI sentences are included. Thus, the core test set consists of 192 utterances and the MIT development set consists of 400 utterances. Each utterance is accompanied with a time-aligned phonetic transcription.

We mapped the 64 phonetic labels in the transcription into 39 phones as in [Lee and Hon, 1989]. From each utterance we extracted a sequence of feature vectors using a 25 ms Hamming window and a window shift of 10 ms. Each feature vector consisted of 13 Mel-frequency cepstral coefficients, including the 0th cepstral coefficient, augmented with their delta and acceleration coefficients. This resulted in 39-dimensional vectors.

One HMM was estimated for each of the 39 phones. Each model was a left-to-right HMM with 3 states, and with a Gaussian mixture model (GMM) with 10 mixture components in each state having diagonal covariance matrices. The models were trained using the maximum likelihood (ML) criterion on all segments in the TIMIT labels having at least 3 feature vectors. This was the case for 128806 segments. Also in the development set and the test set, only the segments with at least 3 feature vectors were used. The development set consisted of 13918 segments and the test set of 6587 segments. The ML trained generative models resulted in 70.6% accuracy on the TIMIT core test set.

Penalized logistic regression with adaptive regressor parameters

Logistic regression using the likelihood mapping with adaptive regressor parameters was used. Only the mean values of the HMMs were updated. The initial weight matrix $\boldsymbol{W}$ was estimated using the penalized logistic regression machine (PLRM) with 10 Newton iterations, each having 2000 conjugate gradient iterations. The RProp method with 100 iterations was used to update the HMM mean values. The update of $\boldsymbol{W}$ in each coordinate descent iteration was done using PLRM with 2 Newton iterations, each having 2000 conjugate gradient iterations. The MIT development set was

used to select a CD iteration for which to stop the training algorithm, and to select a value for the regularization parameter δ. The values found were 3 CD iterations and $\delta = 1000$, which resulted in a test accuracy of 79.9%. Table 4.5 summarizes the results. It can be seen that PLR with adaptive regressor parameters clearly outperforms the baseline system.

Table 4.5: Accuracy on the TIMIT core test set using PLR with the likelihood mapping and adaptive regressor parameters.

Baseline	PLR
70.6%	79.9%

We would like to compare our results using PLR with the results obtained with hidden conditional random fields (HCRF) in [Gunawardana et al., 2005], and the conditional augmented (C-Aug) models in [Layton and Gales, 2006]. However, in those papers, the authors didn't use the segmentation provided with the TIMIT database, but rather estimated the phone boundaries using forced alignment segmentation. In this way, they could make use of all the segments in the labellings. These account for 140225 training segments, 15057 development segments, and 7215 test segments, which are 11419, 1139, and 628 fewer segments, respectively, than our approach. Nevertheless, we present their results in Table 4.6. The baseline system and the HCRF consisted of HMMs with 10 mixture components, and their accuracies were reported in [Gunawardana et al., 2005]. Later, they improved their optimization algorithm and obtained an accuracy of 78.7% with 20 mixture components [Mahajan et al., 2006]. The C-Aug models referred to in Table 4.6 contained HMMs with 10 mixture components, and the accuracy of this system was reported in [Layton and Gales, 2006]. Although the results in tables 4.5 and 4.6 cannot be directly compared, one observation that may favor the PLR classifier over the other two, is that PLR has a larger improvement from its baseline than HCRF and the C-Aug model.

Table 4.6: Accuracy on the TIMIT core test set using HCRF and C-Aug models.

Baseline	HCRF	The C-Aug model
71.9%	78.2%	76.6%

4.4 Summary and Discussion

This chapter has presented methods for isolated-word speech recognition using logistic regression. An essential element of this approach is the mapping function from variable length vector sequences into fixed-dimensional vectors. Various explicit mappings were presented, including the likelihood mapping and the likelihood-ratio mapping. Mappings that are implicitly defined through kernel functions were also presented. Of particular interest is the global alignment (GA) kernel which is a sequence kernel that operates directly on pairs of feature vector sequences extracted from speech signals. These mappings and kernels can be used directly in the logistic regression framework, thereby allowing classification of words spoken in isolation. The approach is not restricted to words only, but can be used for classification of speech segments into other linguistic units such as phones. This is an important feature, since it allows us to extend the approach to continuous speech recognition. This is the topic of the next chapter.

A thorough analysis was presented on the task of recognizing the English letters in the E-set spoken in isolation. Penalized logistic regression with the likelihood mapping and adaptive regressor parameters was found to achieve the highest accuracy. Also, preliminary experiments with the GA kernel showed promising performance and deserves further study. An experiment on the task of phone classification was also presented. The approach with the likelihood mapping and adaptive regressor parameters achieved an accuracy which is comparable with the performance of today's state-of-the-art methods on this particular problem.

Chapter 5

N-best Rescoring using Logistic Regression on Segments

In the previous chapter we explored ways of using logistic regression in isolated-word speech recognition. The conditional probability of a word given a speech segment was obtained. Although we only considered probabilistic prediction of words given a speech segment, the theory is directly applicable to subword units such as phones. In this chapter, we consider the *continuous speech recognition* problem, which amounts to finding the best *sequence of subwords*, or *sentence hypothesis*, given a whole utterance of a sentence. A problem we have to deal with in this context is that the segment boundaries are not known. We propose a two step approach: 1) generate an N-best list using a set of hidden Markov models and the Viterbi algorithm, and 2) rescore the N-best list and select the sentence hypothesis with the highest score. Rescoring of a sentence hypothesis is done by obtaining probabilities of each subword using logistic regression, and combining the subword probabilities into a new sentence score using a geometric mean. These sentence scores can either be used directly to reorder the sentence hypotheses in the N-best list, or they can be interpolated with the HMM likelihood of the corresponding sentence hypotheses before reordering. The recognized sentence hypothesis of an utterance is then taken to be the first one in the N-best list, i.e., the sentence hypothesis with the highest score.

The outline of the chapter is as follows. In the next section, we introduce some notation, and consider an approach to continuous speech recognition that uses only the segmentation information provided by the Viterbi algorithm to relabel and rescore a sentence hypothesis (1-best list). In Section

5.2 we present our way of rescoring N-best lists. Experimental results on connected digits recognition using the Aurora2 database, and continuous phone recognition using the TIMIT database are presented in Section 5.3. Finally, Section 5.4 contains a short summary and a discussion.

Part of the work presented in this chapter is based on [Birkenes et al., 2006b, 2007].

5.1 Relabeling and Rescoring Sentence Hypotheses

In the following, let us assume that we have a set of hidden Markov models (HMMs), one for each subword (e.g., a digit in a spoken digit string, or a phone). We will refer to these HMMs as the baseline models and they will play an important role in both the training phase and the test phase of our proposed approach for connected speech recognition using logistic regression. For convenience, we let $\boldsymbol{Z}$ denote a sequence of feature vectors extracted from a spoken utterance of a sentence $\boldsymbol{s} = (y^{(1)}, \ldots, y^{(L_{\boldsymbol{s}})})$ with $L_{\boldsymbol{s}}$ subwords. Each subword label $y^{(l)}$ is one of $\{1, \ldots, C\}$, where C denotes the number of different subwords. Given a feature vector sequence $\boldsymbol{Z}$ extracted from a spoken utterance $\boldsymbol{s}$, the baseline models can be used in conjunction with the Viterbi algorithm in order to generate a sentence hypothesis $\hat{\boldsymbol{s}} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(L_{\hat{s}})})$, which is a hypothesized sequence of subwords. Additional information provided by the Viterbi algorithm are the maximum likelihood (ML) segmentation on the subword level, and approximations to the subword likelihoods. We write the ML segmentation as $\boldsymbol{Z} = (\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(L_{\hat{s}})})$, where $\boldsymbol{X}^{(l)}$ denotes a sequence of feature vectors associated with the lth subword $\hat{y}^{(l)}$ of the sentence hypothesis. To summarize, there are three types of information provided by the Viterbi algorithm; the sentence hypothesis, its ML segmentation, and the Viterbi-approximated subword likelihoods. These pieces of information can be used in various ways in our system for connected speech recognition using logistic regression.

Perhaps the simplest way is to make use of only the segmentation information, and use logistic regression on each segment $\boldsymbol{X}^{(l)}$ in order to obtain a new subword label $\hat{y}^{(l)}$, possibly different from the one in the sentence hypothesis, along with its estimated conditional probability $\hat{p}_{\hat{y}^{(l)}}$. The concatenation of the new subword labels gives rise to a new sentence hypothesis $\hat{\boldsymbol{s}} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(L_{\hat{s}})})$. Moreover, the geometric mean of the subword probabilities may serve as a confidence measure $p_{\hat{\boldsymbol{s}}}$ for the sentence hypothesis,

that is,

$$p_{\hat{\boldsymbol{s}}} = \left(\prod_{l=1}^{L_{\hat{\boldsymbol{s}}}} p_{\hat{y}^{(l)}} \right)^{1/L_{\hat{\boldsymbol{s}}}}. \tag{5.1}$$

In order to do the above relabeling and rescoring of a sentence hypothesis, the logistic regression model must be trained on the segments of the training data. If the training utterances were segmented on the subword level, i.e., if we knew the segment boundaries of each subword, we could simply use these subword-labeled segments as the training set for the logistic regression model. In most training databases for speech however, the segment boundaries are not known, only the orthographic transcription, i.e., the correct subword sequence. Then, the most straightforward thing to do would be to estimate the segment boundaries. For this, we will make use of the baseline models to perform Viterbi forced alignment (FA) segmentation on the training data. From a pair $(\boldsymbol{Z}, \boldsymbol{s})$ in the training database, FA segmentation gives us a set $\{(\boldsymbol{X}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{X}^{(L_{\boldsymbol{s}})}, y^{(L_{\boldsymbol{s}})})\}$ of subword labeled segments. Doing this for all the pairs $(\boldsymbol{Z}, \boldsymbol{s})$ in the training database yields a set

$$\mathcal{D}_{\text{FA}} = \{(\boldsymbol{X}^{(l)}, y^{(l)})\}_{l=1,\ldots,L_{\text{FA}}} \tag{5.2}$$

of all FA-labeled segments. This set can then be used in the training of the logistic regression model using one of the methods in Chapter 4.

In the testing phase, the above procedure may work well if the segmentation of each test utterance provided by the Viterbi algorithm is close to the true unknown segmentation. However, the procedure is likely to fail if the segmentation contains segments that are incorrect, since such incorrect segments have not been seen by the training algorithm. Their domain typically lies outside the domain for correct segments, and the logistic regression model therefore tends to give very high probability to one subword (often incorrect) and correspondingly low probability to the others. In the next section, we generate a set of sentence hypotheses with corresponding segmentation in the form of an N-best list, in the hope that at least one of the hypotheses and its corresponding segmentation is correct or nearly correct. Our approach will be to rescore each sentence hypothesis using subword probabilities obtained from logistic regression and select the sentence hypothesis with the highest score as the recognition output.

5.2 Rescoring N-Best Lists

For a given utterance we can use the baseline models to generate an N-best list of the N most likely sentence hypotheses. An example of a 5-best list is

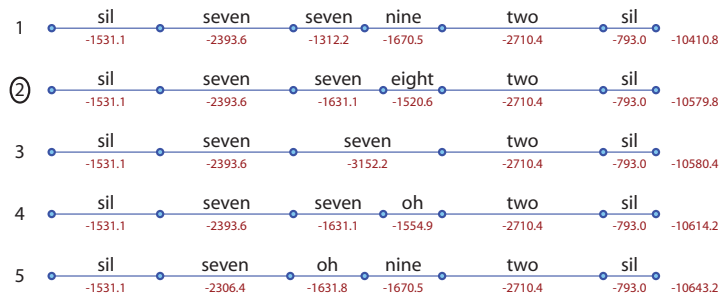

Figure 5.1: A 5-best list where the numbers below the arcs are Viterbi-approximated log-likelihood values corresponding to the segments. The total log-likelihood for each sentence hypothesis is shown at the right. The list is sorted after decreasing log-likelihood values for the sentences.

shown in Figure 5.1. The list is generated for an utterance of the sentence "seven, seven, eight, two", with leading and trailing silence. The most likely sentence hypothesis according to the HMMs appears at the top of the list and is the sentence "seven, seven, nine, two". This sentence differs from the correct sentence, which is the second most likely sentence hypothesis, by one subword. The segmentation of each sentence hypothesis in the list is the most likely segmentation given the sentence hypothesis. Each segment is accompanied with the Viterbi-approximated log-likelihood.

The reason for generating N-best lists is to obtain a set of sentence hypotheses with different labeling and corresponding segmentation, from which the best sentence hypothesis can be chosen based on additional knowledge. In the following we will first consider how we can obtain reliable subword probabilities given speech segments appearing in N-best lists. We suggest to use a garbage class for this purpose. Then, we introduce a method for rescoring N-best lists using these estimated subword probabilities.

5.2.1 Logistic Regression on Segments in N-best Lists

Provided that the baseline models are reasonably good, many of the segments in the N-best lists are correct in the sense that they correspond to a complete utterance of exactly one subword. However, it is inherent that N-best lists frequently contain segments that do not correspond to a complete utterance of exactly one subword. Some segments, for example, correspond to only a part of an utterance of a subword, or even an utterance of several subwords together. Consider again the 5-best list in Figure 5.1, where the correct sentence hypothesis appears in position 2. Let us assume that the correct unknown segmentation coincides with the ML segmentation in

position 2. Then, the third segment in sentence hypothesis 3 actually corresponds to an utterance of the two connected digits "seven" and "eight" spoken in a sequence. Moreover, for hypotheses 1 and 5, the third segment may not correspond to a complete utterance of "seven", while the fourth segment corresponds to an utterance of the last part of "seven" and the whole of "eight". Thus, the segments of an N-best list can be roughly divided into two parts: *correct segments* and *garbage segments*.

The role of logistic regression in our N-best rescoring approach is to provide conditional probabilities of subword labels given a segment. Obviously, we want a correct subword label to get high conditional probability given a correct segment. This implies that incorrect subword labels will get low probabilities for correct segments since the total probability should sum to one. Furthermore, garbage segments should result in low probabilities for all subword labels. As was noted in the previous section, the latter is likely to fail if such garbage segments have not been used in the training phase.

For this reason we introduce a *garbage class*, whose role is to aggregate large probability for garbage segments and low probability otherwise. In the training of the logistic regression model, we therefore need two sets of training examples; 1) a set of correct segments each labeled with the correct subword label, and 2) a set of garbage segments labeled with the garbage label. The former set is taken to be the set $\mathcal{D}_{\text{FA}}$ in (5.2). Extracting garbage segments to be used in the training of the logistic regression model is more difficult. In the rescoring phase, segments that differ somehow from the true unknown segments should give small probability to any class in the vocabulary, and therefore high probability to the garbage class. In order to achieve this, we generate an N-best list for each training utterance, and compare all segments within the list with the corresponding forced alignment generated segments, or the true segments if they are known. The segments from the N-best list that have at least ϵ number of frames not in common with any of the forced alignment segments, are labeled with the garbage label $C+1$ and used as garbage segments for training. This gives us a set

$$\mathcal{D}_{\text{gar}} = \{(\boldsymbol{X}^{(l)}, C+1)\}_{l=1,\ldots,L_{\text{gar}}} \tag{5.3}$$

of all garbage-labeled segments. The full training data used to train the logistic regression model is therefore

$$\mathcal{D} = \mathcal{D}_{\text{FA}} \cup \mathcal{D}_{\text{gar}}. \tag{5.4}$$

The training is done using one of the methods presented in Chapter 4.

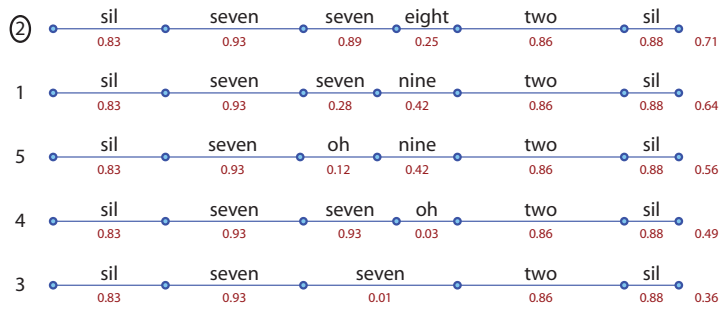

Figure 5.2: The 5-best list in Figure 5.1 after rescoring using penalized logistic regression with HMM likelihoods as regressors and adaptive regressor parameters. The hypotheses have been reordered according to sentence probabilities computed from geometric means of the segment probabilities.

5.2.2 The Rescoring Procedure

Now that we have seen how logistic regression can be used to obtain the conditional probability of a subword given a segment, we will see how we can use these probability estimates to rescore and reorder sentence hypotheses of an N-best list. Two approaches are presented. First we present our method for rescoring N-best lists when a garbage class is used. Then, we briefly review the rescoring approach taken in [Ganapathiraju et al., 2004; Birkenes et al., 2006b] where no garbage class was used.

For a given sentence hypothesis $\hat{\boldsymbol{s}} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(L_{\hat{s}})})$ in an N-best list with corresponding segmentation $\boldsymbol{Z} = (\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(L_{\hat{s}})})$, we can use logistic regression to compute the conditional probabilities $p_{\hat{y}^{(l)}} = p(y = \hat{y}^{(l)} | \boldsymbol{X}^{(l)}, \boldsymbol{W})$. A score for the sentence hypothesis can then be taken as the geometric mean of these probabilities as in (5.1). In order to avoid underflow errors caused by multiplying a large number of small numbers, the log of the score above is usually computed instead, that is,

$$\log p_{\hat{\boldsymbol{s}}} = \frac{1}{L_{\hat{\boldsymbol{s}}}} \sum_{l=1}^{L_{\hat{\boldsymbol{s}}}} \log p_{\hat{y}^{(l)}}. \tag{5.5}$$

When all hypotheses in the N-best list have been rescored with the probabilistic score above, they can be reordered in descending order based on their new score. Figure 5.2 shows the 5-best list in 5.1 after rescoring and reordering. Now, the correct sentence hypothesis "seven, seven, eight, two" has the highest score and is on top of the list.

Alternatively, for a given sentence hypothesis $\hat{\boldsymbol{s}}$, the score in (5.1) can be interpolated with the log-likelihood value for the sentence hypothesis

provided by the Viterbi algorithm. Let $\tilde{p}(\hat{\boldsymbol{s}}|\boldsymbol{Z})$ denote the posterior sentence probability that can in theory be obtained from the sentence HMM likelihood $\tilde{p}(\boldsymbol{Z}|\hat{\boldsymbol{s}})$. The log of the weighted geometric mean with weight $0 \leq \alpha \leq 1$ between the two sentence scores can then be written as

$$S_{\hat{s}} = (1-\alpha)\log \hat{p}_{\hat{\boldsymbol{s}}} + \alpha \log \tilde{p}(\hat{\boldsymbol{s}}|\boldsymbol{Z}) \tag{5.6}$$

$$\propto (1-\alpha)\log \hat{p}_{\hat{\boldsymbol{s}}} + \alpha(\log \tilde{p}(\boldsymbol{Z}|\hat{\boldsymbol{s}}) + \log \tilde{p}(\hat{\boldsymbol{s}})), \tag{5.7}$$

since $\tilde{p}(\boldsymbol{Z})$ is constant for all hypotheses in an N-best list. Furthermore, since we have assumed that $\tilde{p}(\hat{s})$ is constant we can write

$$S_{\hat{s}} \propto (1-\alpha)\log \hat{p}_{\hat{\boldsymbol{s}}} + \alpha \log \tilde{p}(\boldsymbol{Z}|\hat{\boldsymbol{s}}). \tag{5.8}$$

The right-hand side of the above equation is taken as the interpolated score. Note that if $\alpha = 0$, only the logistic regression score is used for rescoring, while if $\alpha = 1$, only the HMM score is used.

Rescoring without a garbage class

If the logistic regression model is trained without a garbage class and without seeing garbage segments, rescoring of N-best lists is likely to produce unreliable results. One way to alleviate this problem is to try to limit rescoring to only those sentence hypotheses that are likely not to contain garbage segments that differ significantly from the true unknown segments.

The approach taken in [Ganapathiraju et al., 2004; Birkenes et al., 2006b] was to limit rescoring to only those sentence hypotheses that have the same number of subwords as the first sentence hypothesis in the N-best list. In this way, the correct sentence cannot be found if the first hypothesis in the N-best list does not contain the correct number of subwords. Nevertheless, the approach has the potential of correcting substitution errors generated by the baseline system, in particular hypotheses that have correct or approximately correct segmentation, but incorrect subword labels. For the 5-best list in Figure 5.1, only sentence hypotheses number 1, 2, 4, and 5 would be rescored using this approach, with sentence hypothesis number 3 being omitted since it has one less segment than the hypothesis at the top of the list.

5.3 Experiments

In this section we present experimental results on continuous speech recognition. First we present results on connected digit recognition using the Aurora2 database. Then, we present results on continuous phone recognition using the TIMIT database.

5.3.1 Connected Digit Recognition using the Aurora2 Database

Various experiments were conducted. First, we tried the relabeling approach which amounts to recognizing spoken digit strings using logistic regression on the segments provided by the baseline HMM recognizer. Only the segmentation information from the baseline recognizer was used in this approach. Next, we performed rescoring of 5-best lists generated by the baseline recognizer. We tried both rescoring without a garbage class (the approach in [Ganapathiraju et al., 2004; Birkenes et al., 2006b]) and with a garbage class. In these experiments we used information of both segmentation and the corresponding subword labels. Finally, we did one experiment where we interpolated the logistic regression score and the HMM score, and thereby used all the information in the 5-best lists provided by the baseline recognizer.

The Aurora2 database and the baseline system

The Aurora2 connected digits database [Pearce and Hirsch, 2000] contains utterances, from different speakers, of digit strings with lengths 1–7 digits. We used only the clean data in both training and testing. The clean data corresponds to the data in the TI-digits database [Leonard, 1984] downsampled to 8 kHz and filtered with a G712 characteristic. There are 8440 training utterances and 4004 test utterances in the training set and the test set, respectively. The speakers in the test set are different from the speakers in the training set.

From each speech signal, a sequence of feature vectors were extracted using a 25 ms Hamming window and a window shift of 10 ms. Each feature vector consisted of 13 Mel-frequency cepstral coefficients (MFCC) and the frame energy, augmented with their delta and acceleration coefficients. This resulted in 39-dimensional vectors.

Each of the digits 1–9 was associated with one class, while 0 was associated with two classes reflecting the pronunciations “zero” and “oh”. The number of digit classes was thus $C = 11$. For each of the 11 digit classes, we used an HMM with 16 states and 3 mixtures per state. In addition, we used a silence (sil) model with 3 states and 6 mixtures per state, and a short pause (sp) model with 1 state and 6 mixtures. These HMM topologies are the same as the ones defined in the training script distributed with the database. We refer to these models as the baseline models, or collectively as the baseline recognition system. The sentence accuracy on the test set using the baseline system was 96.85%.

Relabeling sentence hypotheses

We used penalized logistic regression (PLR) with the likelihood mapping and adaptive regressor parameters. For the regressors we used one HMM for each digit, with the same topology as the baseline models. There were $C = 11$ digit classes in the logistic regression model.

Before training the logistic regression model, the training data was segmented using the baseline models with forced alignment. The segmentation resulted in 27727 segments. We updated only the mean values of the HMMs while keeping the other HMM parameters fixed. For each coordinate descent iteration we iterated 5 times in the steepest descent method and 3 times in the Newton method. At each step in the steepest descent method, the stepsize was either doubled or halved, depending on whether the previous step resulted in a decreased or an increased value of the criterion.

In the testing phase, we first used the baseline system to recognize each utterance in the test set. Then, logistic regression was used to classify each of the segments of each sentence hypothesis. Finally, for each utterance, the new digit labels were concatenated to form a new sentence label. The resulting sentence accuracy was 97.20%, only a slight improvement from the baseline system which achieved 96.85% sentence accuracy.

Rescoring 5-best lists without a garbage class

Next we used the already trained logistic regression model to rescore 5-best lists that were generated on the test set by the baseline recognition system. The upper bound on the sentence accuracy inherent in the 5-best lists, i.e., the sentence accuracy obtainable with a perfect rescoring method, was 99.18%. We chose to rescore only those sentence hypotheses in each 5-best list that had the same number of digits as the first hypothesis in the list. The same approach was taken in [Ganapathiraju et al., 2004], where the authors rescored N-best lists using the support vector machine (SVM), and in [Birkenes et al., 2006b]. The resulting sentence accuracy was 97.20%, which is the same as the sentence accuracy achieved by the relabeling approach. Table 5.1 summarizes the results.

Table 5.1: Sentence accuracy of the baseline system, the relabeling approach, and the 5-best rescoring approach without a garbage class.

Baseline	5-best upper bound	1-best relabeling	5-best rescoring
96.85%	99.18%	97.20%	97.20%

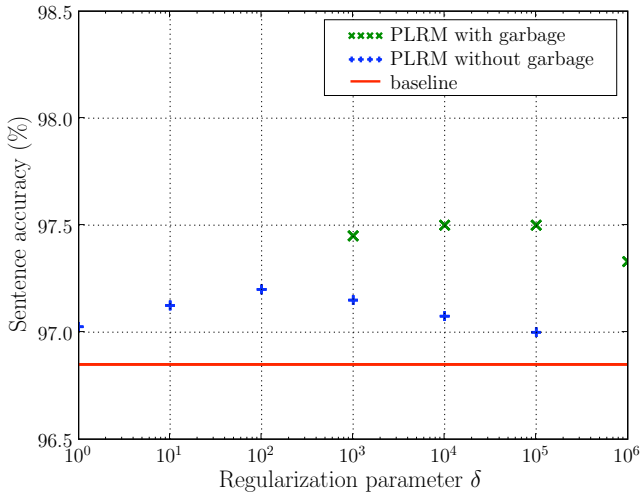

Figure 5.3: Sentence accuracy on the test set for various δ.

Rescoring 5-best lists with a garbage class

In the following, we present results that we achieved with 5-best rescoring with the use of a garbage class in the logistic regression model. We also used an extra class for silence, as well as two extra regressors (HMMs) for garbage and silence, respectively. For the garbage and silence HMMs we used 16 states and 3 mixtures per state. The 5-best lists used in the rescoring phase were the same as above. Training was done using two sets of segments; correct segments with the correct class label, and garbage segments with the garbage label. The former set was generated using the baseline recognition system with forced alignment on the training data. This resulted in 44607 segments. The garbage segments were generated from 5-best lists on the training data, with $\epsilon = 10$. This method resulted in 22380 garbage segments. In total, there were 66987 training segments. Again, we updated only the mean values of the HMMs while keeping the other HMM parameters fixed. For each coordinate descent iteration we used the RProp method [Riedmiller and Braun, 1993] with 100 iterations and initial stepsize 0.01, as well as 4 Newton iterations. After 30 coordinate descent iterations, the optimization was stopped due to time limitations.

The sentence accuracies for $\delta \in \{10^3, 10^4, 10^5, 10^6\}$ are shown in Figure 5.3. The baseline accuracy and the accuracy of the 5-best rescoring approach without a garbage class are also shown. We see that our approach with a garbage class gives the best accuracy for the four values of the regularization parameter δ we used in our experiments. For lower values of δ,

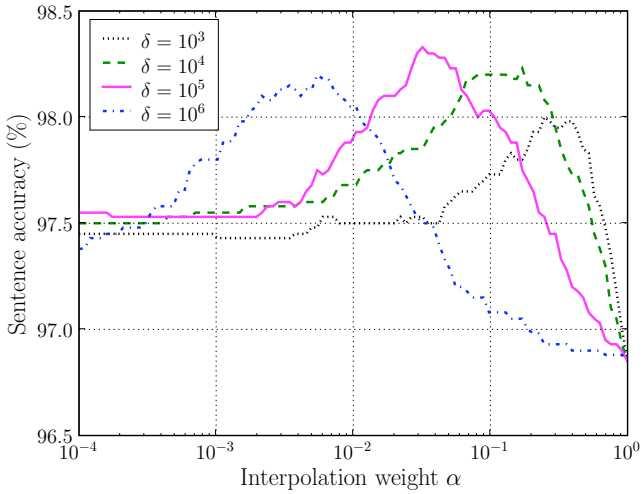

Figure 5.4: Sentence accuracy using interpolated scores.

we expect a somewhat lower sentence accuracy due to over-fitting. Very large δ values are expected to degrade the accuracy since the regression likelihood will be gradually negligible compared to the penalty term.

Figure 5.4 shows the effect of interpolating the HMM sentence likelihood with the logistic regression score. Note that with $\alpha = 0$, only the logistic regression score is used in the rescoring, and when $\alpha = 1$, only the HMM likelihood is used. The large gain in performance when taking both scores into account can be explained by the observation that the HMM score and the logistic regression score made very different sets of errors.

5.3.2 Continuous Phone Recognition on the TIMIT Database

We did experiments on continuous phone recognition on TIMIT [Lamel et al., 1986]. A two-step approach was taken: 1) generate N-best lists using a high performance baseline speech recognizer, and 2) rescore the N-best lists using logistic regression with a garbage class. Each regressor was a HMM-based detector for 15 broad classes of manner and place of articulation, namely fricative, vowel, stop, nasal, semi-vowel, low, mid, high, labial, coronal, dental, velar, glottal, retroflex, and silence. The output of each detector was the frame-normalized log-likelihood ratio between a model and an antimodel. Thus, the mapping used in logistic regression was the likelihood-ratio mapping with the 15 articulatory classes. The purpose for the experiment was twofold. First, in order to test our rescoring ap-

proach on phones, and second to investigate the use of penalized logistic regression in the new paradigm of knowledge-based speech recognition [Lee, 2004; Siniscalchi et al., 2006].

The TIMIT database and the baseline system

Information on the TIMIT database can be found in Section 4.3.2. We mapped the 64 phonetic labels in the TIMIT transcriptions into 39 phones [Lee and Hon, 1989], and ignored the glottal stop [Halberstadt and Glass, 1997]. As a baseline system we used a high performance phone recognizer developed at the Brno University of Technology (BUT) [Schwarz et al., 2006]. The BUT recognizer is a hybrid ANN/HMM system, where each HMM has three states with a single Gaussian mixture component in each state. For more details about the BUT system, the reader is referred to [Schwarz et al., 2006]. The BUT recognizer was used in order to generate forced alignment segmentation of the training set, as well as for generation of N-best lists for both training and testing.

Rescoring of 5-best and 20-best lists on TIMIT

Forced alignment segmentation on the phone level of the training set was performed with the baseline recognizer. Each of the 124931 phone segments was then labeled with one or more articulatory classes and used in the ML training of the 15 target HMMs and their corresponding anti-models. We used 3 states with 16 mixtures components per state for each of the 30 HMMs. The phone segments generated with forced alignment were also used in order to train the logistic regression model with adaptive parameters. In addition, we used a set of garbage segments that were generated from a 100-best list on the training data using the baseline recognizer. We used $\epsilon = 3$, which provided us with 36016 garbage segments. In total there were thus 160947 segments in the training data. Only the mean values of the target HMMs were updated, while all the other HMM parameters were kept fixed. For each of the coordinate descent iteration, we used the RProp method [Riedmiller and Braun, 1993] with 100 iterations to update the HMM means $\boldsymbol{\lambda}$ and the Newton method with 4 iterations to update the weight matrix W. After 6 coordinate descent iterations, the optimization was stopped due to time limitations.

Rescoring was done on both 5-best lists and 20-best lists, both generated on the TIMIT core test set by the BUT baseline system. Table 5.2 lists the baseline performance and the upper bounds for the 5-best lists and the 20-best lists in terms of phone accuracy.

Tables 5.3 and 5.4 summarize the experimental results for several values

Table 5.2: Phone accuracy and upper bounds with 5-best lists and 20-best lists using the BUT baseline recognition system.

Baseline	5-best upper bound	20-best upper bound
74.42	77.00	79.14

of the regularization parameter δ for the 5-best lists and the 20-best lists, respectively. Only the logistic regression scores are displayed, i.e., with the interpolation weight set to $\alpha = 0$.

Table 5.3: Phone accuracy of the 5-best rescoring approach.

$\delta = 10^{-1}$	$\delta = 10$	$\delta = 10^3$
74.66	74.66	74.62

Table 5.4: Phone accuracy of the 20-best rescoring approach.

$\delta = 10^{-1}$	$\delta = 10$	$\delta = 10^3$
75.21	74.57	74.90

By comparing the last rows of Tables 5.3 and 5.4 with the baseline performance, we can see that PLRM achieves better performance for all three values of δ. For the rescoring of the 20-best list with $\delta = 10^{-1}$, we achieve a relative improvement in accuracy of 16.74%.

Figures 5.5 and 5.6 show the effect of interpolating the HMM score with the logistic regression score for the 5-best lists and 20-best lists, respectively, for $\delta \in \{10^{-1}, 10, 10^3\}$. Note that with $\alpha = 0.1$, we obtain accuracies that are close to the results presented in Tables 5.3 and 5.4 for the logistic regression score only, while with $\alpha = 1$, only the HMM baseline score is used, so the baseline performance of 74.42% is obtained.

The following observations are worth noting. First, an increment in the phone accuracy is observed when passing from 5-best lists to 20-best lists, so we expect the performance to improve by increasing the number of competing hypotheses. Second, by combining logistic regression scores and baseline scores there is only a small improvement, which might mean that the two systems make the same type of errors.

A final important observation is that rescoring with PLRM generates more deletion errors than the baseline. The improved accuracies are caused by a decrease in substitution and insertion errors.

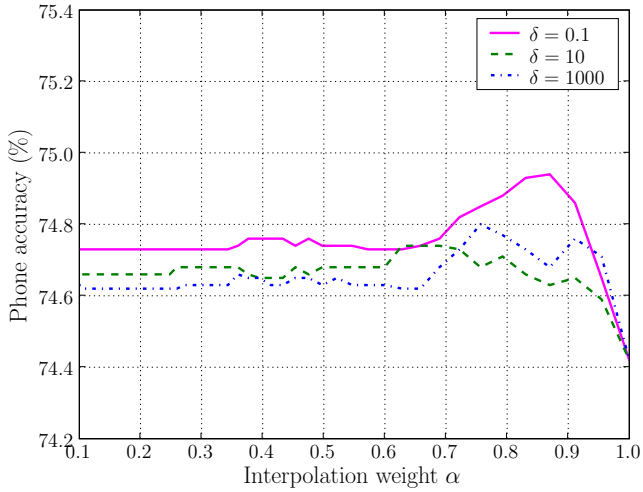

Figure 5.5: Phone accuracy using interpolated scores for the 5-best list.

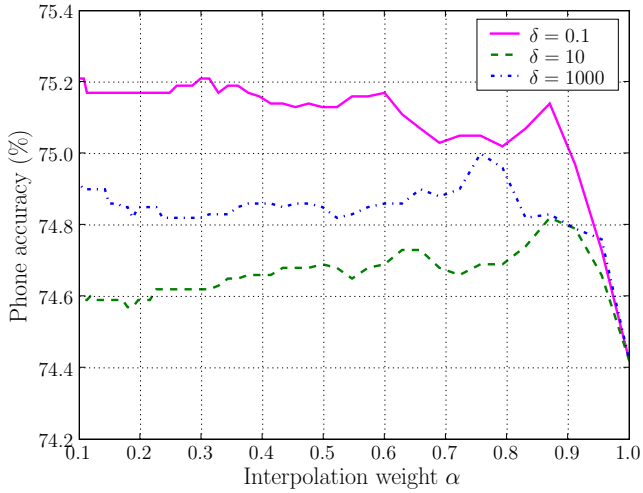

Figure 5.6: Phone accuracy using interpolated scores for the 20-best list.

5.4 Summary and Discussion

A two-step approach to continuous speech recognition using logistic regression on speech segments has been presented. In the first step, a set of hidden Markov models (HMMs) is used in conjunction with the Viterbi algorithm in order to generate an N-best list of sentence hypotheses for the utterance to be recognized. In the second step, each sentence hypothesis is rescored by interpolating the HMM sentence score with a new sentence score obtained by combining subword probabilities provided by a logistic regression model. We argued that a logistic regression model with a garbage class is necessary in this approach.

Results from two sets of experiments were presented. The first set of experiments concerned connected digit recognition. Among the methods we tried, the approach with a garbage class achieved the highest sentence accuracy. Moreover, combining the HMM sentence score with the logistic regression score showed significant improvements in accuracy. A likely reason for the large improvement is that the HMM baseline approach and the logistic regression approach generated different sets of errors.

The second set of experiments were conducted on the TIMIT continuous phone recognition task. A high performance discriminative baseline system was used to generate 5-best lists and 20-best lists. A set of 15 HMM likelihood-ratio detectors for place and manner of articulation were used as regressors. We achieved improvements in phone accuracy for all three values of the regularization parameter δ that we tried, for both the 5-best lists and the 20-best lists. The interpolation with the HMM scores did not result in significant improvements. The reason for this may be that both the baseline system and the logistic regression approach made the same types of errors. Moreover, in going from 5-best lists to 20-best lists the overall accuracy was increased, suggesting that a larger hypothesis space will result in even higher performance.

In both sets of experiments, the improved overall accuracies were due to a decrease in the number of substitution errors and insertion errors compared to the baseline system. The number of deletion errors, however, increased compared to the baseline system. A possible reason for this may be the difficulty of sufficiently covering the space of long garbage segments in the training phase of the logistic regression model. This is an issue that needs further study.

Chapter 6

Conclusions and Future Work

We have presented a framework for automatic speech recognition using logistic regression. First, we gave a thorough overview of the logistic regression framework, including *penalized logistic regression* (PLR) and its dual formulation which is known as *kernel logistic regression* (KLR). Then, we explored ways of using logistic regression for isolated-word speech recognition, followed by extensions to the more difficult problem of continuous speech recognition.

For isolated-word speech recognition we dealt with the difficulty of variable-length speech signals by introducing a mapping from feature vector sequences (time series) to fixed-dimensional vectors in the logistic regression framework. Two explicit mappings constructed from a set of *hidden Markov models* (HMMs) were used, namely the *likelihood mapping* and the *likelihood-ratio mapping*. The two mappings were shown to have similar performance on an isolated-word speech recognition task. Joint optimization of the logistic regression weights and the HMM (regressor) parameters was proposed. A penalized logistic likelihood was used as the criterion for the joint optimization. Experiments on the English E-set of the TI46 database improved the accuracy from 88.3% obtained on the baseline conventional generative approach to 96.3% obtained on the proposed combined generative-discriminative approach. Moreover, on phone classification using the TIMIT database, the phone accuracy increased from 70.6% to 79.9%.

We also presented the *global alignment (GA) kernel*, which is a sequence kernel that operates directly on pairs of time series. The use of such a kernel in KLR implies an implicitly defined mapping from time series to fixed-dimensional vectors. Unlike previous sequence kernels for

speech recognition, which has been defined as the score along the optimal alignment path between two time series, the GA kernel is defined as the sum of the scores of all alignment paths. Preliminary results on the TI46 E-set achieved an accuracy of 93.2%. The approach with sequence kernels is a purely discriminative approach, since there are no generative models (HMMs) involved. Although the results are not as good as the combined generative-discriminative approach, we believe that the GA kernel still has room for improvement.

For continuous speech recognition using logistic regression, we dealt with the sequence labeling problem by using a two-step approach. In the first step, a set of HMMs is used to generate an N-best list of sentence hypotheses for a spoken utterance. In the second step, these sentence hypotheses are rescored with the use of logistic regression to obtain a subword probability for each segment. The subword probabilities are then combined to form new sentence scores. These sentence scores are either used directly to reorder the sentence hypotheses in the N-best list, or they are interpolated with the HMM likelihood of the corresponding sentence hypotheses before reordering. We argued that a *garbage class* is necessary in this approach in order to get reliable probability estimates. The two-step approach was first tested on a connected-digit recognition task. The baseline conventional approach achieved a sentence accuracy of 96.9%. Rescoring 5-best lists with logistic regression achieved a slight improvement to 97.5% without interpolation with the HMM likelihood, and a more significant improvement to 98.3% when interpolation was done. These results were obtained using optimal values of the regularization parameter δ and interpolation weight α. The upper bound on the rescoring accuracy of the 5-best list was 99.2%. The approach was also tested on a continuous phone recognition task. A high-performance baseline system was used, with a phone accuracy of 74.4%. Rescoring 20-best lists using logistic regression resulted in a phone accuracy of 75.2% with optimal δ and α. The upper bound on the rescoring accuracy of the 20-best lists was 79.1%.

6.1 Future Work

In this section we give some suggestions for future work. We start with a discussion on future work for PLR with adaptive regressors, then KLR with a sequence kernel, and finally continuous speech recognition using logistic regression.

Penalized logistic regression with adaptive regressor parameters

We presented two mappings to be used for speech recognition, the likelihood mapping and the likelihood-ratio mapping. They both contained average log-likelihood per frame, with the length of the speech segments normalized out. This was done to ensure that two utterances of the same subword spoken with different speaking rate would map to the same region in regressor space. There is no doubt, however, that segment lengths also contain important discriminative information. Consider for example the discrimination between utterances of the digits "oh" and "seven", which typically have very different durations. This is an issue that needs further attention.

In this thesis, we did not consider the use of gradients of the HMMs with respect to the parameters as regressors. This is an interesting research direction since it has been shown [Jaakkola and Haussler, 1999b; Smith and Gales, 2002] that the gradient contains important discriminative information. One of the difficulties with this approach, however, is the large dimension of the regressor (gradient) space.

The HMMs are important in our combined generative-discriminative approach. In this thesis, the HMM parameters were initially trained using the maximum likelihood (ML) criterion. Then they were updated by maximizing a penalized logistic regression criterion, which contains many local maxima. It would be interesting to see in what degree a different set of initial HMM parameters would affect the recognition error. In particular, models trained with maximum mutual information (MMI) [Bahl et al., 1986] or minimum classification error (MCE) [Juang et al., 1997] could be attempted.

The logistic regression framework is flexible in the sense that the regressors may contain arbitrary information about the speech segment to be classified. In this thesis, we only used HMM-based likelihoods, where all HMMs were trained using the same feature extraction method. A direction for future work is to search for other segment-based features that can be added to the set of regressors for improved performance.

The approach with adaptive regressor parameters lead to problems with overfitting to the training data. A penalty on the regressor parameters was proposed, but experiments showed that it did not work well. If a better penalty can be found such that overfitting is avoided, we could use a more efficient optimization method for estimating the weight matrix $\boldsymbol{W}$ and the regressor parameters $\boldsymbol{\lambda}$.

Determining the regularization parameter δ is an important issue in the logistic regression framework. We presented the use of a Bayesian information criterion (ABIC) [Akaike, 1980] to estimate δ for penalized

logistic regression with fixed regressor parameters. This approach could also be attempted for use with KLR or PLR with adaptive regressor parameters.

A major issue in speech recognition that we did not consider in this thesis is the robustness to mismatched conditions (e.g., noise). This is a very important issue for practical applications of speech recognition. The likelihood mapping is not suited for mismatched conditions. This was pointed out in [Smith and Gales, 2002], and our preliminary experiments also showed that it did not work well in noise. The likelihood-ratio mapping is believed to be more robust. Experiments should be conducted in order to verify this. Future research related to robustness include the search for other robust regressors, and adaptation of the parameters $\boldsymbol{W}$ and $\boldsymbol{\lambda}$ to the new conditions.

Kernel logistic regression with sequence kernels

We presented a new sequence kernel for speech which we call the global alignment (GA) kernel. We believe that KLR with the GA kernel has good potential for further improvements. In the preliminary experiments presented in this thesis, only 13-dimensional MFCC feature vectors were used. It would be interesting to see whether the inclusion of delta and acceleration coefficients would improve the performance

An important feature that is missing in the GA kernel is sequence length normalization. This was not included due to difficulties in proving positive definiteness, and needs further study.

It can also be attempted to experiment with other constraints on the allowable alignments and other path weights. A good review of various alternatives of these quantities can be found in [Rabiner and Juang, 1993] for the use in dynamic time warping (DTW).

An important issue with the use of KLR is how to handle large amounts of training data. Recent papers [Krishnapuram et al., 2005; Zhu and Hastie, 2001, 2005; Myrvoll and Matsui, 2006] have presented sparse approximation methods for KLR. It would be interesting to try these methods for KLR with the GA kernel for speech applications.

It remains to be seen how well the GA kernel works in mismatched conditions. A recent paper [Solera-Ureña et al., 2007] demonstrates that the dynamic time-alignment kernel (DTAK) [Shimodaira et al., 2002] performs very well in mismatched conditions. Since the GA kernel shares many similarities with the DTAK, this may indicate that also the GA kernel would work well in mismatched conditions. This should be verified through experiments.

Continuous speech recognition using logistic regression

In our two-step approach to continuous speech recognition, the garbage class plays an important role. For good rescoring performance, it is important to use a representative set of garbage features in the training phase. We attempted this with the use of N-best lists that were generated on the training set. In the rescoring phase, we reported problems with a large number of deletion errors. The reason for this is not fully clear, but it might be caused by the difficulty of covering the large input space of segments that are "too long". This issue needs further study, and other methods for generating garbage segments should be sought for.

Interpolation of the logistic regression score and the HMM sentence score was shown to increase performance. We used linear interpolation for this. Perhaps a nonlinear interpolation would result in even better performance.

Only rescoring of N-best lists were considered in this thesis. A natural extension is to attempt rescoring of lattices instead of N-best lists.

The two-step approach to continuous speech recognition is dependent on a good set of baseline models that can generate good segmentations. If the N-best lists do not contain good segmentations, the performance cannot be expected to be good. An arguably more elegant solution to the continuous speech recognition problem would be to avoid the rescoring paradigm altogether and rather perform the recognition in one step. This is a topic for future research.

Appendix A

Proofs of Lemmas

A.1 Proof of Lemma 2.1.1

Before giving the proof of Lemma 2.1.1, we will introduce a lemma that we will need.

Lemma A.1.1 *The gradient of* $p_c = p(y = c|\boldsymbol{X}, \boldsymbol{W})$ *in eq.* (2.10) *with respect to the parameter for class* d *is*

$$\nabla_{\boldsymbol{w}_d} p_c = p_c([c = d] - p_d)\boldsymbol{\phi}, \tag{A.1}$$

where $[c = d]$ *is* 1 *if* $c = d$ *and* 0 *otherwise, and* $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda})$.

Proof *(Lemma A.1.1)* Using the product rule and the chain rule gives us

$$\begin{aligned}
\nabla_{\boldsymbol{w}_d} \boldsymbol{p}_c &= \nabla_{\boldsymbol{w}_d} \frac{e^{\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{\phi}}}{\sum_{i=1}^{C} e^{\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\phi}}} \\
&= -\frac{1}{(\sum_{i=1}^{C} e^{\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\phi}})^2} \left(\sum_{j=1}^{C} e^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\phi}} \nabla_{\boldsymbol{w}_d} \boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\phi} \right) e^{\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{\phi}} \\
&\quad + \frac{1}{\sum_{i=1}^{C} e^{\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\phi}}} e^{\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{\phi}} \nabla_{\boldsymbol{w}_d} \boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{\phi} \\
&= -p_c \sum_{j=1}^{C} p_j \nabla_{\boldsymbol{w}_d} \boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\phi} + p_c \nabla_{\boldsymbol{w}_d} \boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{\phi}.
\end{aligned} \tag{A.2}$$

Then, since $\nabla_{\boldsymbol{w}_d} \boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\phi} = \boldsymbol{\phi}$ when $j = d$ and 0 otherwise, we get

$$\nabla_{\boldsymbol{w}_d} p_c = -p_c p_d \boldsymbol{\phi} + p_c[c = d]\boldsymbol{\phi} \tag{A.3}$$

$$= p_c([c = d] - p_d)\boldsymbol{\phi}. \tag{A.4}$$

∎

Proof *(Lemma 2.1.1)* Let $p^{(n)}_{y^{(n)}} = p(y = y^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{W})$. The gradient of (2.23) satisfies

$$\nabla_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}; \mathcal{D}) = -\sum_{n=1}^{N} \nabla_{\boldsymbol{W}} \log p^{(n)}_{y^{(n)}} + \frac{\delta}{2} \nabla_{\boldsymbol{W}} \operatorname{trace} \boldsymbol{\Gamma} \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{W}. \tag{A.5}$$

For the first gradient in (A.5), we can apply the chain rule to get

$$\nabla_{\boldsymbol{W}} \log p^{(n)}_{y^{(n)}} = \frac{1}{p^{(n)}_{y^{(n)}}} \nabla_{\boldsymbol{W}} p^{(n)}_{y^{(n)}}. \tag{A.6}$$

Furthermore, since $\nabla_{\boldsymbol{W}} p^{(n)}_{y^{(n)}} = [\nabla_{\boldsymbol{w}_1} p^{(n)}_{y^{(n)}}, \ldots, \nabla_{\boldsymbol{w}_C} p^{(n)}_{y^{(n)}}]$, we can use Lemma A.1.1 which gives us

$$\nabla_{\boldsymbol{W}} p^{(n)}_{y^{(n)}} = \left[p^{(n)}_{y^{(n)}}([y^{(n)} = 1] - p^{(n)}_1)\boldsymbol{\phi}^{(n)}, \ldots, p^{(n)}_{y^{(n)}}([y^{(n)} = C] - p^{(n)}_C)\boldsymbol{\phi}^{(n)}\right] \tag{A.7}$$

$$= p^{(n)}_{y^{(n)}} \boldsymbol{\phi}^{(n)} (\boldsymbol{e}^{\mathrm{T}}_{y^{(n)}} - \boldsymbol{p}^{(n)\mathrm{T}}). \tag{A.8}$$

Substituting this into (A.6) leads to

$$\nabla_{\boldsymbol{W}} \log p^{(n)}_{y^{(n)}} = \boldsymbol{\phi}^{(n)} (\boldsymbol{e}^{\mathrm{T}}_{y^{(n)}} - \boldsymbol{p}^{(n)\mathrm{T}}). \tag{A.9}$$

For the second gradient in (A.5) we get

$$\nabla_{\boldsymbol{W}} \operatorname{trace} \boldsymbol{\Gamma} \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{W} = \sum_{c=1}^{C} \gamma_c \nabla_{\boldsymbol{W}} \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{w}_c \tag{A.10}$$

$$= \sum_{c=1}^{C} \gamma_c \begin{bmatrix} | & & | & & | \\ 0 & \cdots & 2\boldsymbol{\Sigma}\boldsymbol{w}_c & \cdots & 0 \\ | & & | & & | \end{bmatrix} \tag{A.11}$$

$$= 2\boldsymbol{\Sigma}\boldsymbol{W}\boldsymbol{\Gamma}. \tag{A.12}$$

Finally, substituting (A.9) and (A.12) into (A.5) yields

$$\nabla_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}; \mathcal{D}) = -\sum_{n=1}^{N} \boldsymbol{\phi}^{(n)} (\boldsymbol{e}^{\mathrm{T}}_{y^{(n)}} - \boldsymbol{p}^{(n)\mathrm{T}}) + \frac{\delta}{2} 2\boldsymbol{\Sigma}\boldsymbol{W}\boldsymbol{\Gamma} \tag{A.13}$$

$$= \Phi(\boldsymbol{P}(\boldsymbol{W})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{W}\boldsymbol{\Gamma}. \tag{A.14}$$

∎

A.2 Proof of Lemma 2.1.2

Proof *(Lemma 2.1.2)* Let $\boldsymbol{g}$ denote the vectorization of the gradient (2.24), i.e.,

$$\begin{aligned}
\boldsymbol{g} &= \operatorname{vec} \nabla_{\boldsymbol{W}} \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}; \mathcal{D}) && \text{(A.15)}\\
&= \operatorname{vec} \boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta \operatorname{vec} \boldsymbol{\Sigma} \boldsymbol{W} \boldsymbol{\Gamma} && \text{(A.16)}\\
&= \operatorname{vec} \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)} (\boldsymbol{p}^{(n)\mathrm{T}} - \boldsymbol{e}_{y^{(n)}}^{\mathrm{T}}) + \delta (\boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}) \operatorname{vec} \boldsymbol{W} && \text{(A.17)}\\
&= \sum_{n=1}^{N} \operatorname{vec} \boldsymbol{\phi}^{(n)} (\boldsymbol{p}^{(n)\mathrm{T}} - \boldsymbol{e}_{y^{(n)}}^{\mathrm{T}}) + \delta (\boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}) \operatorname{vec} \boldsymbol{W} && \text{(A.18)}\\
&= \sum_{n=1}^{N} (\boldsymbol{p}^{(n)} - \boldsymbol{e}_{y^{(n)}}) \otimes \boldsymbol{\phi}^{(n)} + \delta (\boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}) \operatorname{vec} \boldsymbol{W}. && \text{(A.19)}
\end{aligned}$$

The transpose of the gradient is needed in order to find the Hessian. It is

$$\boldsymbol{g}^{\mathrm{T}} = \sum_{n=1}^{N} (\boldsymbol{p}^{(n)\mathrm{T}} - \boldsymbol{e}_{y^{(n)}}^{\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)\mathrm{T}} + \delta \operatorname{vec} \boldsymbol{W}^{\mathrm{T}} (\boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}). \tag{A.20}$$

Let the gradient operator with respect to $\operatorname{vec} \boldsymbol{W}$ be denoted $\nabla_{\vec{\boldsymbol{W}}}$. Then, the Hessian of (2.23) satisfies

$$\begin{aligned}
\nabla_{\vec{\boldsymbol{W}}}^2 \mathcal{P}_{\delta}^{\log}(\boldsymbol{W}; \mathcal{D}) &= \nabla_{\vec{\boldsymbol{W}}} \boldsymbol{g}^{\mathrm{T}} && \text{(A.21)}\\
&= \sum_{n=1}^{N} \nabla_{\vec{\boldsymbol{W}}} (\boldsymbol{p}^{(n)\mathrm{T}} \otimes \boldsymbol{\phi}^{(n)\mathrm{T}}) + \delta \nabla_{\vec{\boldsymbol{W}}} \operatorname{vec} \boldsymbol{W}^{\mathrm{T}} (\boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}) && \text{(A.22)}\\
&= \sum_{n=1}^{N} (\nabla_{\vec{\boldsymbol{W}}} \boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)\mathrm{T}} + \delta \boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}. && \text{(A.23)}
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
\nabla_{\vec{\boldsymbol{W}}} \boldsymbol{p}^{(n)\mathrm{T}} &= \begin{bmatrix} \nabla_{\boldsymbol{w}_1} p_1^{(n)} & \cdots & \nabla_{\boldsymbol{w}_1} p_C^{(n)} \\ \vdots & & \vdots \\ \nabla_{\boldsymbol{w}_C} p_1^{(n)} & \cdots & \nabla_{\boldsymbol{w}_C} p_C^{(n)} \end{bmatrix} && \text{(A.24)}\\
&= \begin{bmatrix} p_1^{(n)}(1 - p_1^{(n)}) \boldsymbol{\phi}^{(n)} & \cdots & -p_C^{(n)} p_1^{(n)} \boldsymbol{\phi}^{(n)} \\ \vdots & & \vdots \\ p_1^{(n)} p_C^{(n)} \boldsymbol{\phi}^{(n)} & \cdots & p_C^{(n)}(1 - p_C^{(n)}) \boldsymbol{\phi}^{(n)} \end{bmatrix} && \text{(A.25)}\\
&= (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)}. && \text{(A.26)}
\end{aligned}$$

Substituting this expression into (A.23) leads to

$$\nabla^2_{\boldsymbol{W}} \mathcal{P}^{\log}_{\delta}(\boldsymbol{W}; \mathcal{D}) = \sum_{n=1}^{N} ((\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)}) \otimes \boldsymbol{\phi}^{(n)\mathrm{T}} + \delta \boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma} \tag{A.27}$$

$$= \sum_{n=1}^{N} (\operatorname{diag} \boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)} \boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)} \boldsymbol{\phi}^{(n)\mathrm{T}} + \delta \boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}. \tag{A.28}$$

∎

A.3 Proof of Lemma 2.1.4

Before giving the proof of Lemma 2.1.4, we will introduce a lemma that we will need.

Lemma A.3.1 *The gradient of $p_c = p(y = c|\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\lambda})$ in eq. (2.10) with respect to the lth element of $\boldsymbol{\lambda}$ is*

$$\frac{\partial}{\partial \lambda_l} p_c = p_c (\boldsymbol{w}_c^{\mathrm{T}} - \boldsymbol{p}^T \boldsymbol{W}^T) \frac{\partial}{\partial \lambda_l} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}). \tag{A.29}$$

Proof *(Lemma A.3.1)* Using the product rule and the chain rule gives us

$$\begin{aligned}
\frac{\partial}{\partial \lambda_l} p_c &= \frac{\partial}{\partial \lambda_l} \frac{e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}}{\sum_{i=1}^{C} e^{\boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}} \\
&= -\frac{1}{(\sum_{i=1}^{C} e^{\boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})})^2} \left(\sum_{j=1}^{C} e^{\boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})} \frac{\partial}{\partial \lambda_l} \boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}) \right) e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})} \\
&\quad + \frac{1}{\sum_{i=1}^{C} e^{\boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})}} e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X};\boldsymbol{\lambda})} \frac{\partial}{\partial \lambda_l} \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}) \\
&= -p_c \sum_{j=1}^{C} p_j \frac{\partial}{\partial \lambda_l} \boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}) + p_c \frac{\partial}{\partial \lambda_l} \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}) \\
&= p_c \left(\boldsymbol{w}_c^{\mathrm{T}} - \sum_{j=1}^{C} p_j \boldsymbol{w}_j^{\mathrm{T}} \right) \frac{\partial}{\partial \lambda_l} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}) \\
&= p_c (\boldsymbol{w}_c^{\mathrm{T}} - \boldsymbol{p}^T \boldsymbol{W}^T) \frac{\partial}{\partial \lambda_l} \boldsymbol{\phi}(\boldsymbol{X}; \boldsymbol{\lambda}).
\end{aligned} \tag{A.30}$$

∎

Proof *(Lemma 2.1.4)* Let $p^{(n)}_{y^{(n)}} = p(y = y^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{W}, \boldsymbol{\lambda})$. The partial derivative of (2.63) satisfies

$$\begin{aligned}\frac{\partial}{\partial \lambda_l}\mathcal{P}^{\log}_{\delta}(\boldsymbol{W}, \boldsymbol{\lambda}, \mathcal{D}) = &-\sum_{n=1}^{N}\frac{\partial}{\partial \lambda_l}\log p^{(n)}_{y^{(n)}} \\ &+\frac{\delta}{2}\frac{\partial}{\partial \lambda_l}\operatorname{trace}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}(\boldsymbol{\lambda})\boldsymbol{W}.\end{aligned} \tag{A.31}$$

For the first part of (A.31), we can apply the chain rule to get

$$-\sum_{n=1}^{N}\frac{\partial}{\partial \lambda_l}\log p^{(n)}_{y^{(n)}} = -\sum_{n=1}^{N}\frac{1}{p^{(n)}_{y^{(n)}}}\frac{\partial}{\partial \lambda_l}p^{(n)}_{y^{(n)}} \tag{A.32}$$

Furthermore, we can use Lemma 2.1.4 which gives us

$$\frac{\partial}{\partial \lambda_l}p^{(n)}_{y^{(n)}} = p^{(n)}_{y^{(n)}}\left(\boldsymbol{w}^{\mathrm{T}}_{y^{(n)}} - \boldsymbol{p}^{(n)\mathrm{T}}\boldsymbol{W}^{\mathrm{T}}\right)\frac{\partial}{\partial \lambda_l}\boldsymbol{\phi}(\boldsymbol{X}^{(n)}; \boldsymbol{\lambda}) \tag{A.33}$$

Substituting this into (A.32) leads to

$$-\sum_{n=1}^{N}\frac{\partial}{\partial \lambda_l}\log p^{(n)}_{y^{(n)}} = -\sum_{n=1}^{N}\left(\boldsymbol{w}^{\mathrm{T}}_{y^{(n)}} - \boldsymbol{p}^{(n)\mathrm{T}}\boldsymbol{W}^{\mathrm{T}}\right)\frac{\partial}{\partial \lambda_l}\boldsymbol{\phi}(\boldsymbol{X}^{(n)}; \boldsymbol{\lambda}) \tag{A.34}$$

$$= -\sum_{n=1}^{N}\left(\boldsymbol{e}^{\mathrm{T}}_{y^{(n)}}\boldsymbol{W}^{T} - \boldsymbol{p}^{(n)\mathrm{T}}\boldsymbol{W}^{\mathrm{T}}\right)\frac{\partial}{\partial \lambda_l}\boldsymbol{\phi}(\boldsymbol{X}^{(n)}; \boldsymbol{\lambda}) \tag{A.35}$$

$$= -\sum_{n=1}^{N}\left(\boldsymbol{e}^{\mathrm{T}}_{y^{(n)}} - \boldsymbol{p}^{(n)\mathrm{T}}\right)\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial \lambda_l}\boldsymbol{\phi}(\boldsymbol{X}^{(n)}; \boldsymbol{\lambda}) \tag{A.36}$$

$$= -\operatorname{trace}\left\{\left(\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{P}^{\mathrm{T}}(\boldsymbol{W})\right)\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial \lambda_l}\boldsymbol{\Phi}\right\} \tag{A.37}$$

$$= \operatorname{trace}\left\{\left(\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{W}) - \boldsymbol{Y}^{\mathrm{T}}\right)\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial \lambda_l}\boldsymbol{\Phi}\right\}. \tag{A.38}$$

For the second partial derivative in (A.31) we get

$$\frac{\partial}{\partial\lambda_l}\operatorname{trace}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}(\boldsymbol{\lambda})\boldsymbol{W} = \frac{\partial}{\partial\lambda_l}\operatorname{trace}\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\frac{1}{N}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \tag{A.39}$$

$$= \frac{\partial}{\partial\lambda_l}\operatorname{trace}\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda}) \tag{A.40}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial}{\partial\lambda_l}\operatorname{trace}\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda}) \tag{A.41}$$

$$= \frac{2}{N}\sum_{n=1}^{N}\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda})\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial\lambda_l}\boldsymbol{\phi}(\boldsymbol{X}^{(n)};\boldsymbol{\lambda}) \tag{A.42}$$

$$= \frac{2}{N}\operatorname{trace}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial\lambda_l}\boldsymbol{\Phi}, \tag{A.43}$$

where the second last equality stems from use of the chain rule and the fact that $\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}$ is symmetric.

Finally, substituting (A.34) and (A.43) into (A.31) yields

$$\begin{aligned}\frac{\partial}{\partial\lambda_l}\mathcal{P}_\delta^{\log}(\boldsymbol{W},\boldsymbol{\lambda};\mathcal{D}) &= \operatorname{trace}\left\{\left(\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{W})-\boldsymbol{Y}^{T}\right)\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial\lambda_l}\boldsymbol{\Phi}\right\}\\ &\quad+\frac{\delta}{N}\operatorname{trace}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial\lambda_l}\boldsymbol{\Phi}\\ &= \operatorname{trace}\left\{\left(\frac{\delta}{N}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{\Gamma}+\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{W})-\boldsymbol{Y}^{\mathrm{T}}\right)\boldsymbol{W}^{\mathrm{T}}\frac{\partial}{\partial\lambda_l}\boldsymbol{\Phi}\right\}.\end{aligned} \tag{A.44}$$

∎

A.4 Proof of Lemma 2.2.1

Proof Since we know the gradient of $\mathcal{P}_\delta^{\log}(\boldsymbol{W};\mathcal{D})$ with respect to $\boldsymbol{W}$ and since $\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\boldsymbol{V}$, we can use the chain rule to find the gradient of $\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D})$. The chain rule can be written

$$\operatorname{vec}\nabla_{\boldsymbol{V}}\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}V}\operatorname{vec}\nabla_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W};\mathcal{D}), \tag{A.45}$$

where $\mathrm{d}\boldsymbol{W}/\mathrm{d}V$ is the $NC \times (M+1)C$ Jacobian matrix containing the partial derivatives of all elements of $\boldsymbol{W}$ with respect to all elements of $\boldsymbol{V}$, and can be found to be

$$\frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}\boldsymbol{V}} = \boldsymbol{I}_C \otimes \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}. \tag{A.46}$$

Substituting $\mathrm{d}\boldsymbol{W}/\mathrm{d}V$ and $\mathcal{P}_\delta^{\log}(\boldsymbol{W};\mathcal{D})$ into (A.45) leads to

$$\operatorname{vec}\nabla_{\boldsymbol{V}}\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \boldsymbol{I}_C \otimes \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} \operatorname{vec}\{\boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Sigma}\boldsymbol{W}\boldsymbol{\Gamma}\} \tag{A.47}$$

$$= \operatorname{vec}\{\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{\Gamma}\}. \tag{A.48}$$

Then

$$\nabla_{\boldsymbol{V}}\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}(\boldsymbol{P}(\boldsymbol{W})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{\Gamma}, \tag{A.49}$$

and after substitution of $\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\boldsymbol{V}$ and $\boldsymbol{P}(\boldsymbol{W}) = \check{\boldsymbol{P}}(\boldsymbol{V})$ we get

$$\nabla_{\boldsymbol{V}}\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}(\check{\boldsymbol{P}}(\boldsymbol{V})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}}) + \delta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\boldsymbol{V}\boldsymbol{\Gamma}. \tag{A.50}$$

Finally we use the definition of the kernel matrix $\boldsymbol{K} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}$ and get

$$\nabla_{\boldsymbol{V}}\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \boldsymbol{K}(\check{\boldsymbol{P}}(\boldsymbol{V})^{\mathrm{T}} - \boldsymbol{Y}^{\mathrm{T}} + \delta\boldsymbol{V}\boldsymbol{\Gamma}). \tag{A.51}$$

∎

A.5 Proof of Lemma 2.2.2

Proof Using the chain rule and the product rule we can obtain the following formula for calculating the Hessian matrix:

$$\nabla^2_{\boldsymbol{V}}\check{\mathcal{P}}_\delta^{\log}(\boldsymbol{V};\mathcal{D}) = \frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}\boldsymbol{V}}\nabla^2_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W};\mathcal{D})\left(\frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}\boldsymbol{V}}\right)^{\mathrm{T}} + \operatorname{vec}\nabla_{\boldsymbol{W}}\mathcal{P}_\delta^{\log}(\boldsymbol{W};\mathcal{D}) \otimes \frac{\mathrm{d}^2\boldsymbol{W}}{\mathrm{d}\boldsymbol{V}^2}, \tag{A.52}$$

where $\mathrm{d}\boldsymbol{W}/\mathrm{d}V$ is the Jacobian in (A.46) and $\mathrm{d}^2\boldsymbol{W}/\mathrm{d}V^2$ denotes the derivative of the Jacobian with respect to $\boldsymbol{V}$. Since the Jacobian is not a function

of $\boldsymbol{V}$, we have that $\mathrm{d}^2\boldsymbol{W}/\mathrm{d}V^2 = 0$ and we can write

$$\begin{aligned}
\nabla^2_{\boldsymbol{V}}\check{\mathcal{P}}^{\log}_{\delta}(\boldsymbol{V};\mathcal{D}) &= \frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}\boldsymbol{V}}\nabla^2_{\boldsymbol{W}}\mathcal{P}^{\log}_{\delta}(\boldsymbol{W};\mathcal{D})\left(\frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}\boldsymbol{V}}\right)^{\mathrm{T}} \\
&= \boldsymbol{I}_C \otimes \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} \\
&\cdot \left(\sum_{n=1}^{N}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}} + \delta\boldsymbol{\Gamma} \otimes \boldsymbol{\Sigma}\right) \\
&\cdot \boldsymbol{I}_C \otimes \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi} \\
&= \sum_{n=1}^{N}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}^{(n)}\boldsymbol{\phi}^{(n)\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi} \\
&\quad + \delta\boldsymbol{\Gamma} \otimes \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi} \\
&= \sum_{n=1}^{N}(\operatorname{diag}\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(n)}\boldsymbol{p}^{(n)\mathrm{T}}) \otimes \boldsymbol{k}(\boldsymbol{X}^{(n)})\boldsymbol{k}^{\mathrm{T}}(\boldsymbol{X}^{(n)}) + \delta\boldsymbol{\Gamma} \otimes \boldsymbol{K},
\end{aligned} \tag{A.53}$$

where in the last step we have substituted $\boldsymbol{K} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}$ and $\boldsymbol{k}(\boldsymbol{X}^{(n)}) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}^{(n)\mathrm{T}}$. ∎

A.6 Proof of Lemma 4.1.1

Proof Let us for simplicity omit the indexes m and n. The partial derivative is

$$\frac{\partial \log\hat{p}(\boldsymbol{X};\boldsymbol{\lambda})}{\partial\tilde{\mu}_{qhd}} = \sum_{t=1}^{T}\frac{\partial}{\partial\tilde{\mu}_{qhd}}\log p(\boldsymbol{x}_t|\hat{q}_t,\boldsymbol{\eta}_{\hat{q}_t}). \tag{A.54}$$

In the sum over t, only the terms that satisfy $\hat{q}_t = q$ will be nonzero. Thus

$$\frac{\partial \log\hat{p}(\boldsymbol{X};\boldsymbol{\lambda})}{\partial\tilde{\mu}_{qhd}} = \sum_{t=1}^{T}\delta(\hat{q}_t - q)\frac{\partial}{\partial\tilde{\mu}_{qhd}}\log p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q) \tag{A.55}$$

$$= \sum_{t=1}^{T}\delta(\hat{q}_t - q)\frac{1}{p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}\frac{\partial}{\partial\tilde{\mu}_{qhd}}p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q). \tag{A.56}$$

Furthermore, in the GMM, only mixture component h will be nonzero giving

$$\frac{\partial p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}{\partial\tilde{\mu}_{qhd}} = c_{qh}(2\pi)^{-D/2}\left(\prod_{d=1}^{D}\sigma_{qhd}\right)^{-1}\frac{\partial}{\partial\tilde{\mu}_{qhd}}e^{-\frac{1}{2}\sum_{i=1}^{D}\left(\frac{x_{ti}-\tilde{\mu}_{qhi}\sigma_{qhi}}{\sigma_{qhi}}\right)^2} \tag{A.57}$$

Using the chain rule we get

$$\frac{\partial p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}{\partial\tilde{\mu}_{qhd}} = c_{qh}(2\pi)^{-D/2}\left(\prod_{d=1}^{D}\sigma_{qhd}\right)^{-1} e^{-\frac{1}{2}\sum_{i=1}^{D}\left(\frac{x_{ti}-\tilde{\mu}_{qhi}\sigma_{qhi}}{\sigma_{qhi}}\right)^2} \left(-\frac{1}{2}\frac{2}{\sigma_{qhd}^2}(x_{td}-\tilde{\mu}_{qhd}\sigma_{qhd})\cdot(-\sigma_{qhd})\right), \quad \text{(A.58)}$$

which reduces to

$$\frac{\partial p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}{\partial\tilde{\mu}_{qhd}} = c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh},\boldsymbol{\Sigma}_{qh})\left(\frac{x_{td}}{\sigma_{qhd}} - \tilde{\mu}_q hd\right) \quad \text{(A.59)}$$

$$= c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh},\boldsymbol{\Sigma}_{qh})\left(\frac{x_{td}-\mu_{qhd}}{\sigma_{qhd}}\right). \quad \text{(A.60)}$$

Substituting this into (A.56) yields

$$\frac{\partial \log\hat{p}(\boldsymbol{X};\boldsymbol{\lambda})}{\partial\tilde{\mu}_{qhd}} = \delta(\hat{q}_t-q)\frac{c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh},\boldsymbol{\Sigma}_{qh})}{p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}\left(\frac{x_{td}-\mu_{qhd}}{\sigma_{qhd}}\right). \quad \text{(A.61)}$$

∎

A.7 Proof of Lemma 4.1.2

Proof Let us for simplicity omit the indexes m and n. Similar to (A.56) we get

$$\frac{\partial \log\hat{p}(\boldsymbol{X};\boldsymbol{\lambda})}{\partial\tilde{\sigma}_{qhd}} = \sum_{t=1}^{T}\delta(\hat{q}_t-q)\frac{1}{p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}\frac{\partial}{\partial\tilde{\sigma}_{qhd}}p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q). \quad \text{(A.62)}$$

Furthermore

$$\frac{\partial p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}{\partial\tilde{\sigma}_{qhd}} = c_{qh}(2\pi)^{-D/2}\frac{\partial}{\partial\tilde{\sigma}_{qhd}}\left(\prod_{i=1}^{D}\sigma_{qhi}\right)^{-1} e^{-\frac{1}{2}\sum_{j=1}^{D}\left(\frac{x_{tj}-\mu_{qhj}\sigma_{qhi}}{e^{\tilde{\sigma}_{qhj}}}\right)^2} \quad \text{(A.63)}$$

$$= c_{qh}(2\pi)^{-D/2}\frac{\partial}{\partial\tilde{\sigma}_{qhd}}\left(e^{-\sum_{i=1}^{D}\left(\tilde{\sigma}_{qhi}+\frac{1}{2}(x_{ti}-\mu_{qhi})^2e^{-2\tilde{\sigma}_{qhi}}\right)}\right). \quad \text{(A.64)}$$

Using the chain rule yields

$$c_{qh}(2\pi)^{-D/2}\left(\prod_{d=1}^{D}\sigma_{qhd}\right)^{-1} e^{-\frac{1}{2}\sum_{i=1}^{D}\left(\frac{x_{ti}-\tilde{\mu}_{qhi}\sigma_{qhi}}{\sigma_{qhi}}\right)^2}\left((x_{td}-\mu_{qhd})^2e^{-2\tilde{\sigma}_{qhd}}-1\right), \quad \text{(A.65)}$$

which reduces to

$$\frac{\partial p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}{\partial \tilde{\sigma}_{qhd}} = c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh},\boldsymbol{\Sigma}_{qh})\Big((x_{td}-\mu_{qhd})^2 e^{-2\tilde{\sigma}_{qhd}} - 1\Big) \tag{A.66}$$

$$= c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh},\boldsymbol{\Sigma}_{qh})\left(\left(\frac{x_{td}-\mu_{qhd}}{\sigma_{qhd}}\right)^2 - 1\right). \tag{A.67}$$

Substituting this into (A.62) yields

$$\frac{\partial \log \hat{p}(\boldsymbol{X};\boldsymbol{\lambda})}{\partial \tilde{\sigma}_{qhd}} = \delta(\hat{q}_t - q)\frac{c_{qh}\mathcal{N}(\boldsymbol{\mu}_{qh},\boldsymbol{\Sigma}_{qh})}{p(\boldsymbol{x}_t|q,\boldsymbol{\eta}_q)}\left(\left(\frac{x_{td}-\mu_{qhd}}{\sigma_{qhd}}\right)^2 - 1\right). \tag{A.68}$$

∎

Appendix B

Estimation of Hidden Markov Model Parameters

We start this appendix with the definition of two variables that are needed for efficient computation of the update equations for the HMM parameters. The variables are known as the *forward variable* and the *backward variable*.

In Chapter 3 we presented an algorithm for computing the likelihood $p(\boldsymbol{X})$ that proceeds forward in time. The algorithm is called the *forward algorithm*, and at each time t it computes the *forward variable*

$$\alpha(q_t) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, q_t), \quad q_t = 1, \ldots, Q. \tag{B.1}$$

There is also an algorithm for computing the likelihood that proceeds backwards in time which is based on the following equation:

$$\begin{aligned} p(\boldsymbol{X}) = \sum_{q_1} \pi_{q_1} p(\boldsymbol{x}_1|q_1) \sum_{q_2} a_{q_1,q_2} p(\boldsymbol{x}_2|q_2) \ldots \\ \cdots \sum_{q_{T-1}} a_{q_{T-2},q_{T-1}} p(\boldsymbol{x}_{T-1}|q_{T-1}) \sum_{q_T} a_{q_{T-1},q_T} p(\boldsymbol{x}_T|q_T). \end{aligned} \tag{B.2}$$

This is the basis for the *backward algorithm.*

Algorithm B.0.1 (The Backward Algorithm) *Computes the likelihood $p(\boldsymbol{X})$ of a time series $\boldsymbol{X}$ with respect to a hidden Markov model.*

1. *Initialize:*

$$\beta(q_T) = 1, \quad q_1 = 1, \ldots, Q \tag{B.3}$$

2. *Iterate: for $t = T-1, \ldots, 1$*

$$\beta(q_t) = \sum_{q_{t+1}} a_{q_t,q_{t+1}} p(\boldsymbol{x}_{t+1}|q_{t+1}) \beta(q_{t+1}), \quad q_t = 1, \ldots, Q \tag{B.4}$$

3. Terminate:

$$p(\boldsymbol{X}) = \sum_{q_1} \pi_{q_1} p(\boldsymbol{x}_1|q_1)\beta(q_1) \tag{B.5}$$

For each time t, the *backward variable* $\beta(q_t)$ is the conditional probability of the partial time series $(\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_T)$ given the state variable q_t, i.e.,

$$\beta(q_t) = p(\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_T|q_t), \quad q_t = 1, \ldots, Q. \tag{B.6}$$

Thus, it can be easily seen from (B.1) and (B.6) that the likelihood of the whole time series $\boldsymbol{X}$ can also be computed as

$$p(\boldsymbol{X}) = \sum_{q_t} \alpha(q_t)\beta(q_t) \tag{B.7}$$

at any time t.

The maximum likelihood estimate of the HMM parameters can be found by maximizing the likelihood, or equivalently, by maximizing the log-likelihood with respect to the HMM parameters. Mathematically, the maximum likelihood estimate using a single observation $\boldsymbol{X}$ is

$$\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}\in\Lambda} \log p(\boldsymbol{X};\boldsymbol{\lambda}) \tag{B.8}$$

$$= \arg\max_{\boldsymbol{\lambda}\in\Lambda} \log \sum_{\boldsymbol{q}} p(\boldsymbol{X},\boldsymbol{q};\boldsymbol{\lambda}), \tag{B.9}$$

where Λ denotes the parameter space of the HMM, i.e., the set of all allowable values for $\boldsymbol{\lambda}$. The above maximization is difficult since the log-likelihood is the logarithm of a sum. Instead of maximizing the log-likelihood itself, we can maximize a lower bound B on the log-likelihood obtained by an application of Jensen's inequality as follows:

$$\log p(\boldsymbol{X};\boldsymbol{\lambda}) = \log \sum_{\boldsymbol{q}} p(\boldsymbol{X},\boldsymbol{q};\boldsymbol{\lambda}) \tag{B.10}$$

$$= \log \sum_{\boldsymbol{q}} f(\boldsymbol{q}|\boldsymbol{X}) \frac{p(\boldsymbol{X},\boldsymbol{q};\boldsymbol{\lambda})}{f(\boldsymbol{q}|\boldsymbol{X})} \tag{B.11}$$

$$\geq \sum_{\boldsymbol{q}} f(\boldsymbol{q}|\boldsymbol{X}) \log \frac{p(\boldsymbol{X},\boldsymbol{q};\boldsymbol{\lambda})}{f(\boldsymbol{q}|\boldsymbol{X})} = B(f,\boldsymbol{\lambda}), \tag{B.12}$$

where $f(\boldsymbol{q}|\boldsymbol{X})$ is a probability distribution over $\boldsymbol{q}$. The *expectation maximization* (EM) algorithm [Dempster et al., 1977; Bilmes, 1997; Jordan, 2007] maximizes the bound $B(f,\boldsymbol{\lambda})$ using coordinate ascent with coordinates f and $\boldsymbol{\lambda}$. Mathematically speaking, the EM algorithm alternately calculates

$$f^{(i+1)} = \arg\max_{f} B(f, \boldsymbol{\lambda}^{(i)}), \tag{B.13}$$

and

$$\boldsymbol{\lambda}^{(i+1)} = \arg\max_{\boldsymbol{\lambda}} B(f^{(i+1)}, \boldsymbol{\lambda}). \tag{B.14}$$

The latter maximization can be simplified by noting that the bound can be written

$$B(f, \boldsymbol{\lambda}) = \sum_{\boldsymbol{q}} f(\boldsymbol{q}|\boldsymbol{X}) \log p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}) - \sum_{\boldsymbol{q}} f(\boldsymbol{q}|\boldsymbol{X}) \log f(\boldsymbol{q}|\boldsymbol{X}) \tag{B.15}$$

$$= E_f \log p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}) - \sum_{\boldsymbol{q}} f(\boldsymbol{q}|\boldsymbol{X}) \log f(\boldsymbol{q}|\boldsymbol{X}), \tag{B.16}$$

where E_f denotes the expectation with respect to the conditional distribution $f(\boldsymbol{q}|\boldsymbol{X})$. Thus, since only the first term is dependent on $\boldsymbol{\lambda}$, (B.14) can be written

$$\boldsymbol{\lambda}^{(i+1)} = \arg\max_{\boldsymbol{\lambda}} E_{f^{(i+1)}} \log p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}). \tag{B.17}$$

The maximum of $B(f, \boldsymbol{\lambda}^{(i)})$ with respect to f occurs when $f = p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)})$. This can be seen from

$$B(p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)}), \boldsymbol{\lambda}^{(i)}) = \sum_{\boldsymbol{q}} p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)}) \log \frac{p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}^{(i)})}{p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)})} \tag{B.18}$$

$$= \sum_{\boldsymbol{q}} p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)}) \log p(\boldsymbol{X}; \boldsymbol{\lambda}^{(i)}) \tag{B.19}$$

$$= \log p(\boldsymbol{X}; \boldsymbol{\lambda}^{(i)}), \tag{B.20}$$

since for this choice, B achieves its maximum which is the log-likelihood function. Thus, the maximization in (B.13) can be written $f^{(i+1)} = p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)})$ and the EM algorithm consists of the following two steps:

E step $\quad \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(i)}) = E_{p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)})} \log p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda})$

M step $\quad \boldsymbol{\lambda}^{(i+1)} = \arg\max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(i)})$.

The E-step amounts to computing the expectation of the *complete log-likelihood* $\log p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda})$ with respect to the conditional distribution $p(\boldsymbol{q}|\boldsymbol{X}; \boldsymbol{\lambda}^{(i)})$. For a hidden Markov model with parameter $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\eta})$ the complete log-likelihood is

$$\log p(\boldsymbol{X}, \boldsymbol{q}; \boldsymbol{\lambda}) = \log \pi_{q_1} + \sum_{t=2}^{T} \log a_{q_{t-1}, q_t} + \sum_{t=1}^{T} \log p(\boldsymbol{x}_t | q_t; \boldsymbol{\eta}_{q_t}). \tag{B.21}$$

Then, taking the expectation with respect to $p(\boldsymbol{q}|\boldsymbol{X};\boldsymbol{\lambda}^{(i)})$ yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda},\boldsymbol{\lambda}^{(i)}) = E_{p(\boldsymbol{q}|\boldsymbol{X};\boldsymbol{\lambda}^{(i)})} \log p(\boldsymbol{X},\boldsymbol{q};\boldsymbol{\lambda}) &= \sum_{\boldsymbol{q}} p(\boldsymbol{q}|\boldsymbol{X};\boldsymbol{\lambda}^{(i)}) \log \pi_{q_1} \\ &+ \sum_{\boldsymbol{q}} p(\boldsymbol{q}|\boldsymbol{X};\boldsymbol{\lambda}^{(i)}) \sum_{t=2}^{T} \log a_{q_{t-1},q_t} \\ &+ \sum_{\boldsymbol{q}} p(\boldsymbol{q}|\boldsymbol{X};\boldsymbol{\lambda}^{(i)}) \sum_{t=1}^{T} \log p(\boldsymbol{x}_t|q_t,\boldsymbol{\eta}).\end{aligned} \tag{B.22}$$

In the expression above $\boldsymbol{\pi}$, $\boldsymbol{A}$ and $\boldsymbol{\eta}$ occur in different terms, which has the nice property that the three components of $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\eta})$ can be optimized independently. Doing so results in the following update equations for the elements of $\boldsymbol{\pi}$ and the elements of $\boldsymbol{A}$:

$$\pi_q^{(i+1)} = p(q_1 = q|\boldsymbol{X};\boldsymbol{\lambda}^{(i)}), \quad q = 1,\ldots,Q, \tag{B.23}$$

$$a_{qr}^{(i+1)} = \frac{\sum_{t=1}^{T} p(q_{t-1} = q, q_t = r|\boldsymbol{X};\boldsymbol{\lambda}^{(i)})}{\sum_{t=1}^{T} p(q_{t-1} = q|\boldsymbol{X};\boldsymbol{\lambda}^{(i)})}, \quad q,r = 1,\ldots,Q. \tag{B.24}$$

The probabilities in the above update equations can be efficiently computed using the forward and backward algorithm. From (B.1) and (B.6) we get

$$p(q_t|\boldsymbol{X}) = \frac{\alpha(q_t)\beta(q_t)}{p(\boldsymbol{X})}, \tag{B.25}$$

and

$$p(q_t, q_{t+1}|\boldsymbol{X}) = \xi(q_t, q_{t+1}) = \frac{\alpha(q_t)p(x_{t+1}|q_{t+1})\beta(q_{t+1})a_{q_t,q_{t+1}}}{p(\boldsymbol{X})}. \tag{B.26}$$

The update equations for the GMM parameters are

$$c_{qh}^{(i+1)} = \frac{\sum_{t=1}^{T} p(q_t = q, h_{q_t,t} = h|\boldsymbol{X},\boldsymbol{\lambda}^{(i)})}{\sum_{t=1}^{T}\sum_{h=1}^{H} p(q_t = q, h_{q_t,t} = h|\boldsymbol{X},\boldsymbol{\lambda}^{(i)})}, \tag{B.27}$$

$$\mu_{qh}^{(i+1)} = \frac{\sum_{t=1}^{T} \boldsymbol{x}_t p(q_t = q, h_{q_t,t} = h|\boldsymbol{X},\boldsymbol{\lambda}^{(i)})}{\sum_{t=1}^{T} p(q_t = q, h_{q_t,t} = h|\boldsymbol{X},\boldsymbol{\lambda}^{(i)})} \tag{B.28}$$

and

$$\Sigma_{qh}^{(i+1)} = \frac{\sum_{t=1}^{T} (\boldsymbol{x}_t - \mu_{qh}^{(i+1)})(\boldsymbol{x}_t - \mu_{qh}^{(i+1)})^{\mathrm{T}} p(q_t = q, h_{q_t,t} = h|\boldsymbol{X},\boldsymbol{\lambda}^{(i)})}{\sum_{t=1}^{T} p(q_t = q, h_{q_t,t} = h|\boldsymbol{X},\boldsymbol{\lambda}^{(i)})}. \tag{B.29}$$

Bibliography

Abou-Moustafa, K., Suen, C., and Cheriet, M. (2004). A generative-discriminative hybrid for sequential data classification. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, Montreal, Quebec, Canada.

Akaike, H. (1980). Likelihood and the bayes procedure. In Bernardo, J., DeGroot, M., Lindley, D., and Smith, A., editors, *Bayesian Statistics*, pages 143–166. Univ. Press, Valencia, Spain.

Albert, A. and Anderson, J. A. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, 71(1):1–10.

Anderson, J. A. and Blair, V. (1982). Penalized maximum likelihood estimation in logistic regression and discrimination. *Biometrika*, 1:123–136.

Aubert, X. L. (2002). An overview of decoding techniques for large vocabulary speech recognition. *Computer Speech and Language*, 16(1):89–114.

Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 49–52.

Bahlmann, C., Haasdonk, B., and Burkhardt, H. (2002). On-line handwriting recognition with support vector machines – a kernel approach. In *Proc. of the 8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 49–54.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statistics*, 41(1):164–171.

Bazzi, I. and Katabi, D. (2000). Using support vector machines for spoken digit recognition. In *6th International Conference on Spoken Language Processing*, Beijing, China.

Berger, J. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, 2 edition.

Bilmes, J. (1997). A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models.

Bilmes, J. (2006). What hmms can do. *IEICE Transactions in Information and Systems*, E89-D(3):869–891.

Birkenes, Ø., Matsui, T., and Tanabe, K. (2005). Probabilistic isolated-word speech recognition via maximum penalized logistic regression likelihood. In *ASJ (Acoustical Society of Japan)*, Sendai, Japan.

Birkenes, Ø., Matsui, T., and Tanabe, K. (2006a). Isolated-word recognition with penalized logistic regression machines. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, Toulouse, France.

Birkenes, Ø., Matsui, T., Tanabe, K., and Myrvoll, T. A. (2006b). Continuous speech recognition with penalized logistic regression machines. In *Proc. IEEE Nordic Signal Processing Symposium*, Reykjavik, Iceland.

Birkenes, Ø., Matsui, T., Tanabe, K., and Myrvoll, T. A. (2007). N-best rescoring for speech recognition using penalized logistic regression machines with garbage class. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, USA.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cuturi, M., Vert, J. P., Birkenes, Ø., and Matsui, T. (2007). A kernel for time series based on global alignments. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, USA.

Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans ASSP*, 28(4):357–366.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *J. Royal Stat. Soc.*, 39(1):1–38.

Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley and Sons, 2nd edition.

Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans ASSP*, 34(1):52–59.

Ganapathiraju, A., Hamaker, J., and Picone, J. (2004). Applications of support vector machines to speech recognition. *IEEE Trans. Signal Processing*, 52(8):2348–2355.

Green, P. and Yandell, B. (1985). Semi-parametric generalized linear models. In Springer, editor, *2nd International GLIM Conference*, number 32 in Lecture notes in Statistics, pages 44–55, Lancaster.

Gunawardana, A., Mahajan, M., Acero, A., and Platt, J. (2005). Hidden conditional random fields for phone classification. In *INTERSPEECH*, pages 1117–1120.

Halberstadt, A. K. and Glass, J. R. (1997). Heterogeneous acoustic measurements for phonetic classification. In *EUROSPEECH*, pages 401–404.

Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436.

Hoerl, A. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67.

Jaakkola, T. and Haussler, D. (1999a). Exploiting generative models in discriminative classifiers. In Solla, S. and D.A.Cohn, editors, *Advances in Neural Information Processing Systems*, pages 487–493. MIT Press.

Jaakkola, T. and Haussler, D. (1999b). Probabilistic kernel regression models. In *Seventh International Workshop on Artificial Intelligence and Statistics*, San Francisco.

Jordan, M. I. (1995). Why the logistic function? a tutorial discussion on probabilities and neural networks. Technical Report 9503, MIT Computational Cognitive Science.

Jordan, M. I. (2007). *An Introduction to Probabilistic Graphical Models*. Book in preparation.

Juang, B.-H., Chou, W., and Lee, C.-H. (1997). Minimum classification error rate methods for speech recognition. *IEEE Trans. Speech and Audio Processing*, 5(3):257–265.

Krishnapuram, B., Carin, L., Figueiredo, M. A. T., and Hartemink, A. J. (2005). Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968.

Lamel, L. F., Kassel, R. H., and Seneff, S. (1986). Speech database development: Design and analysis of the acoustic-phonetic corpus. In Baumann, L. S., editor, *Proc. DARPA Speech Recpgnition Workshop*, pages 100–109.

Layton, M. and Gales, M. (2006). Augmented statistical models for speech recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, Tolouse, France.

Lee, C.-H. (2004). From knowledge-ignorant to knowledge-rich modeling: a new speech research paradigm for next generation automatic speech recognition. In *ICSLP*.

Lee, K. F. and Hon, H. W. (1989). Speaker-independent phone recognition using hidden markov models. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, volume 37, pages 1641–1648.

Leonard, R. (1984). A database for speaker independent digit recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, volume 3, page 42.11.

Luenberger, D. G. (1989). *Linear and Nonlinear Programmin*. Addison-Wesley Publishing Company, 2 edition.

Mahajan, M., Gunawardana, A., , and Acero, A. (2006). Training algorithms for hidden conditional random fields. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, Toulouse, France.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, pages 415–446.

Myrvoll, T. A. and Matsui, T. (2006). On a greedy learning algorithm for dPLRM with applications to phonetic feature detection. In *Proc. IEEE Nordic Signal Processing Symposium*, Reykjavik, Iceland.

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. B. and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14.

Pearce, D. and Hirsch, H.-G. (2000). The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions. In *ISCA ITRW ASR*, pages 181–188, Paris, France.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, L. R. and Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here? *Proc. IEEE*, 88(8):1279–1296.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press.

Schwartz, R. and Chow, Y. (1990). The N-best algorithm: an efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*, volume 1, pages 81–84, Albuquerque, New Mexico, USA.

Schwarz, P., Matějaka, P., and Černocký, J. (2006). Hierarchical structures of neural networks for phoneme recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing (ICASSP)*.

Shimodaira, H., Noma, K., Nakai, M., and Sagayama, S. (2002). Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Information Processing Systems (NIPS) 14*.

Siniscalchi, S., Li, J., and Lee, C.-H. (2006). A study on lattice rescoring with knowledge scores for automatic speech recognition. In *ICSLP*.

Smith, N. and Gales, M. (2002). Speech recognition using SVMs. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 1197–1204. The MIT Press.

Solera-Ureña, R., Martín-Iglesias, D. Gallardo-Antolín, A., Peláez-Moreno, C., and Díaz-de María, F. (2007). Robust ASR using support vector machines. *Speech Communication*.

Tanabe, K. (2001a). Penalized logistic regression machines: New methods for statistical prediction 1. *ISM Cooperative Research Report 143*, pages 163–194.

Tanabe, K. (2001b). Penalized logistic regression machines: New methods for statistical prediction 2. In *Proc. IBIS*, pages 71–76, Tokyo.

Tanabe, K. (2003). Penalized logistic regression machines and related linear numerical algebra. In *KOKYUROKU 1320, Institute for Mathematical Sciences*, pages 239–250, Kyoto.

Ulusoy, I. and Bishop, C. M. (2005). Generative versus discriminative models for object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, 2 edition.

Vert, J.-P., Saigo, H., and Akutsu, T. (2004). Local alignment kernels for biological sequences. In Schölkopf, B., Tsuda, K., and Vert, J.-P., editors, *Kernel Methods in Computational Biology*, pages 131–154. MIT Press.

Viterbi, A. (1983). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, 13(4):179–190.

Williams, C. K. I. and Barber, D. (1998). Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Zhu, J. and Hastie, T. (2001). Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*, volume 14.

Zhu, J. and Hastie, T. (2005). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 1(14):185–205.