Discriminative Training of Hidden Markov Models

Sadik Kapadia

Downing College

March 18, 1998

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where stated. It has not been submitted in whole or part for a degree at any other university.

The length of this thesis including footnotes and appendices does not exceed 25000 words.

Contents

List of Figures

List of Tables

Abstract

Most modern speech recognition systems are based on hidden Markov models. Yet despite their widespread use many of their properties are not well understood. This work aims to increase our understanding about the training of hidden Markov models for classification.

We first examine the question of what is the best measure of hidden Markov model fitness. Our research shows us that the principle that motivated many of the previous studies, the incorrect-model sub-optimality of maximum likelihood, is wrong. We further show that the identity of the best hidden Markov model fitness measure or objective function depends on two outside factors, the model flexibility and the size of the training set. These factors control the point at which training set performance or test set generalisation become the limiting factors. We conjecture that the major effect controlling test set generalisation is the confusion environment in which the state probability density functions are trained. Based on this idea we introduce a new class of hidden Markov model, the frame discriminative hidden Markov model. We focus on zero memory frame discriminative hidden Markov models, these having the same generalisation ability as maximum likelihood hidden Markov models but better training set performance.

We also study the optimisation of the frame discrimination objective function. A comparison of traditional learning techniques with modern machine learning ones shows that the machine learning ones are considerably faster. We also show that it is possible to increase the speed of learning by incorporating extra knowledge about the hessian structure of the fitness surface. Taking these ideas together we obtain a general purpose and reasonably fast training algorithm, on-line Manhattan quick-prop.

We then apply our zero memory frame discriminative objective function and on-line quick-prop to two alphabet recognition tasks. The experimental results provide an empirical verification of the training set/test set performance of frame discriminative training. The results also show that compared to maximum likelihood hidden Markov models, we can produce considerable reductions in model size with frame discriminative hidden Markov models while maintaining the same accuracy.

Lastly we present some ideas for future work.

Keywords: speech recognition,CONNEX,ISOLET,e-set, alphabet, speech recognition, hidden Markov model, maximum likelihood, discriminative training, classification figure of merit, maximum mutual information, frame discrimination, Baum-Welch, optimisation, steepest ascent, conjugate gradients, Newton's algorithm, Manhattan learning, quick-prop.

Abbreviations

CFM Classification figure of merit.

FD Frame discrimination.

HMM Hidden Markov model.

LVQ Learning vector quantisation.

ML Maximum likelihood.

MMI Maximum mutual information.

MMSE Minimum mean square error.

NN Neural network.

PDF Probability density function.

Notation

$\mathcal{I}$ represents an identity matrix.

$[v]_i$ is element i of vector v.

$[m]_{i,j}$ is element (i,j) of matrix m.

v^T is the transpose of vector/matrix v.

$\mathcal{E}(x)$ represents the expected value of x.

$\mathrm{Cov}(x)$ represents the variance of x.

$\mathrm{Cov}(x,y)$ represents the covariance of x and y.

F represents a generic objective function.

$\lambda, \overline{\lambda}, \lambda_0, \ldots, \lambda_k$ each represents different HMM parameter vectors.

n is the size of vectors λ, $\overline{\lambda}$ and $\lambda_0, \ldots, \lambda_k$.

$\frac{\partial F}{\partial \lambda}$ an n by 1 column vector of first derivatives of F.

$\frac{\partial^2 F}{\partial \lambda^2}$ a n by n hessian matrix of F.

U the number of training utterances.

$w(u)$ depending on the context, either the transcription or the synthesised HMM for the u^{th} training utterance. Figure 2.1 illustrates a synthesised training HMM.

O represents the complete set of U preprocessed training utterances.

$O(u)$ are the parameterised speech frames in the u^{th} training utterance.

$O(u)_t$ the speech frame occurring at time t in the u^{th} training utterance.

$T(u)$ is the number of frames in the u^{th} training utterance.

R represents the task's recognition HMM. Figure 2.3 and 2.2 illustrates the recognition HMMs used in this dissertation.

N is the total number of HMM states.

q_i represents state i.

$s \in w$ indicates that s ranges over all possible full state paths in the synthesised HMM w.

$m \in w$ indicates that m ranges over all possible full speech frame/gaussian pairings in the synthesised HMM w.

$s(m, t)$ indexes the speech frame/gaussian pairings m and returns the state at time t. By construction $s(m, 0)$ is the start state for associated HMM.

s_t the state at time t.

$g(m, t)$ indexes the speech frame/gaussian pairings m and returns the gaussian at time t.

g_t the gaussian at time t.

$a_{i,j}$ the transition probability of moving from state i to state j.

K represents the number of gaussians associated with each emitting state.

$m_{i,k}$ represents the mean of gaussian k in state i.

$C_{i,k}$ represents the covariance matrix of gaussian k in state i.

$w_{i,k}$ represents the weight for gaussian k in state i.

$P^*(A|B)$ the true probability of event A given B. B is either an event or HMM.

$P_\lambda(A|B)$ the estimated probability of the event A given B. B is either an event or HMM. λ is the estimator's parameter vector.

Chapter 1

Introduction

The 1990's has seen an explosion of interest in automatic speech transcription. This explosion has many causes including:

1. The availability of cheaper and faster processors.

2. Increased use of computers by the untrained public.

3. Growth of the voice carrying telecommunications network.

The resulting increase in research has led to steady improvements in transcription accuracy [49, 82]. But compared to humans, current recognisers are still fragile and inaccurate.

The ease with which humans can decode speech is deceptive. There are many causes of variability in speech which makes decoding a hard process for computers. The mapping between acoustic signal and transcription is many to many. This non-determinism is due to many factors including:

1. The anatomical build of the speaker. For instance, men have lower fundamental frequencies than women.

2. Dialect differences due to different social and economic backgrounds.

3. Differences in speaking rate.

4. Intonation changes.

5. Alternations in loudness.

6. Grammatical and semantic context of the word.

7. Background sounds.

8. Reverberation.

9. Electronic noise during recording.

10. Homo-phones and word boundary ambiguity.

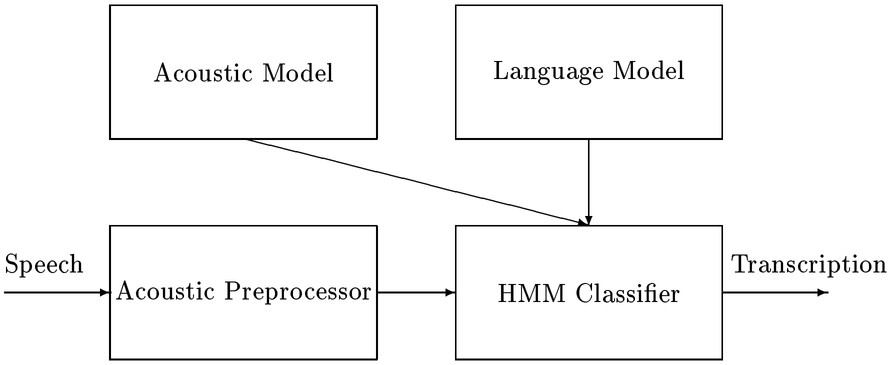

Figure 1.1: Block diagram of our speech recognition system

Given this difficulty it is surprising that the "best" transcription systems to date use a very simple model - a hidden Markov model (HMM).

Figure 1 illustrates the structure of a typical HMM recogniser including the one used in this thesis. It can be broken down into four main components.

The acoustic preprocessor this black-box aims to transform the incoming speech into a form which can be modelled accurately. It also aims to reduce the data rate of the signal and so reduce the computational complexity of the rest of the recognition system. The ideal preprocessor would be able to produce speech frames which contain as much information about the corresponding transcription as does the original waveform.

In theory, given a large amount of training data and a flexible classifier we could model the speech waveform directly and so discard the preprocessor. In practice, the complexity of the classifier is limited by the amount of training data we have [28, 54, 55, 79] and so a good preprocessor is essential. We used standard preprocessors, designed by utilising knowledge about which features of the speech waveform are important for recognition.

There was no attempt made during this dissertation, to improve on these preprocessors. Sections 6.2 and 7.2 describe them in detail.

The acoustic model this black-box categories speech segments. Knowledge is encoded in a model that is known as a HMM. It is described in detail in chapter 2. This work aims to improve this component.

The language model this module imposes grammatical constraints on the transcriptions that are produced by the recogniser. By eliminating impossible transcriptions or down-weighting unlikely transcriptions, the performance of the recogniser can be significantly boosted [24, 45]. This component is discussed in chapter 2.

The classifier this process integrates input from the above components and outputs the best matching transcription. See chapter 2 for more detail.

Before a HMM can be used, its parameters must be determined. This process is commonly known as training. It requires three elements.

1. A training database consisting of speech recordings and their associated transcriptions.

2. An objective function, which together with the training database can be used to measure the HMMs "fitness", and

3. an optimisation procedure which can be used to maximise our objective function.

During training the optimisation procedure is used to search for a HMM parameter vector which has a high fitness. Thus the problem of determining the HMMs parameters is essentially an optimisation process.

In this thesis we shall limit our study to the training of the acoustic model. We assume the rest of the system is given and fixed. In particular, we shall investigate the nature of the objective function, and the methods used for its optimisation.

The standard HMM training process, Maximum Likelihood (ML) training, is described in chapter 2. In chapter 3 we study the properties of ML in some detail. In this chapter, we show that if the HMM is used as a classifier, some of what a ML HMM learns is irrelevant and in fact is no better than noise. In essence a ML HMM makes suboptimal use of it's "regressional flexibility". Also in chapter 3 we shall describe some alternative objective functions, Maximum Mutual Information (MMI) and Frame Discrimination (FD). Compared to ML, MMI and FD make better use of the HMMs "regressional flexibility".

In chapter 4 we present a number of optimisation algorithms that are suitable for use with both MMI and FD. Then chapter 5 compares and contrasts our objective functions against other alternatives to ML that have previously been suggested. We show that many of these alternatives are theoretically suboptimal. Chapters 6 and 7 presents experimental results on two standard speech tasks. These experiments show, that in every case where comparisons were attempted, MMI and FD are better than ML. And in addition, in the vast majority of cases FD HMMs performed better than MMI HMMs. Finally chapter 8 concludes this study by summarising what we have discovered, as well as suggesting avenues for further study.

Chapter 2

Hidden Markov Models

The speech recognition system used in this thesis combines the acoustic and language knowledge sources into one structure — a HMM. This chapter describes the type of HMM that we used.

2.1 Definition

As its name suggests a component part of a HMM is a Markov chain. Markov chains consist of a set of states and a set of transition probabilities. In our system two states are distinguished, the start and end state. All full paths begin in the start state and finish in the end state. The other states are collectively known as the emitting states. Each is associated with a probability density function (PDF). In our case, a mixture of multivariate gaussian distributions.

The popular name hidden Markov model concisely describes the nature of the generating process. We usually see only the output of the PDFs, the corresponding state sequence is hidden. A HMM is a simple (but admittedly not very accurate) model of the speech generation process. We can associate each state with one configuration of the vocal tract. In each state the associated PDF determines the probability that a particular segment of speech signal is produced. The motion of the vocal tract is modelled by the interstate transition probabilities. A complete state path represents a sequence of vocal tract configurations, and by implication a transcription. With this model we can give a figure to the probability that a sequence of parameterised speech vectors came from a particular series of words.

The following have to be specified in order to define a HMM.

1. The number of states N. With no loss of generality the starting state will be numbered 1 and the ending state N.

2. The set of transition probabilities $a_{i,j}$. Where $a_{i,j}$ represents the probability of moving from state i to state j.

3. The mean vectors $m_{i,k}$. Where $m_{i,k}$ represents the mean of gaussian k in state i.

4. The covariance matrices $C_{i,k}$. Where $C_{i,k}$ represents the covariance matrix of gaussian k in state i.

5. The mixing proportions $w_{i,k}$. Where $w_{i,k}$ represents the weight for gaussian k in state i.

By definition these quantities satisfy the following constraints

$$a_{i,j} \geq 0 \quad i,j = 1,\ldots,N \tag{2.1}$$

$$\sum_{j=1}^{N} a_{i,j} = 1 \quad i = 1,\ldots,N \tag{2.2}$$

$$w_{i,k} \geq 0 \quad i = 2,\ldots,N-1, k = 1,\ldots,K \tag{2.3}$$

$$\sum_{k=1}^{K} w_{i,k} = 1 \quad i = 2,\ldots,N-1 \tag{2.4}$$

$$x^T C_{i,k} x > 0 \quad \forall x \neq 0, i = 2,\ldots,N-1, k = 1,\ldots,K \tag{2.5}$$

where K is the number of gaussians associated with each state, and
v^T is the transpose of vector/matrix v.

The joint probability of a state sequence $s_0,\ldots,s_{T(u)},q_N$, (where $s_0 = 1$), and the u^{th} observation sequence $O(u)$ is

$$P_\lambda(s_0,\ldots,s_{T(u)},q_N,O(u)) = P_\lambda(s_0,\ldots,s_{T(u)},q_N)$$

$$P_\lambda(O(u)|s_0,\ldots,s_{T(u)},q_N) \tag{2.6}$$

$$= \left[\prod_{t=0}^{T(u)-1} a_{s_t,s_{t+1}}\right] a_{s_{T(u)},N} \left[\prod_{t=1}^{T(u)} b_{s_t}(O(u)_t)\right] \tag{2.7}$$

where,

$$b_i(O(u)_t) = \sum_{k=1}^{K} w_{i,k} b_{i,k}(O(u)_t) \tag{2.8}$$

$$b_{i,k}(O(u)_t) = \frac{\exp^{-\frac{1}{2}(O(u)_t - m_{i,k})^T C_{i,k}^{-1}(O(u)_t - m_{i,k})}}{(2\pi)^{\frac{n'}{2}} |C_{i,k}|^{\frac{1}{2}}} \tag{2.9}$$

and,
s_t the state at time t.
q_i represents state i.
$\lambda = \{a_{i,j}, m_{i,k}, c_{i,k}, w_{i,k}\}$ is the HMM parameter vector.
$O(u)_t$ is the speech frame occurring at time t in the u^{th} training utterance.
$T(u)$ is the number of frames in the u^{th} training utterance.
n' is the dimension of the speech vector (and by definition the dimension of the means and covariances).

The observant reader will notice at this point that we have used the term probability when sometimes the quantities involved are probability densities. The writer feels this is justified in order to focus attention on other important aspects and to maintain notational compatibility with current literature. Rigorous derivations require a $\delta O(u)_t$ term in 2.9. However these terms cancel in our classification statistic 2.11, leaving our classifier unchanged.

2.2 HMM Modelling Assumptions

As can be seen our model is highly simplified. In particular it makes the following assumptions [10, 13].

1. The probability of the next state is dependent only on the current state. In other words it is independent of the acoustic signal and most of the state sequence.

2. The acoustic signal is dependent only on the current state. However the addition of the delta and delta-delta elements (see sections 6.2 and 7.2), does provide local context dependency.

3. The acoustics are stationary in each state and can be modelled by a mixture of gaussians.

Although these assumptions are incorrect, they are made by many systems in order to reduce the number of parameters that have to be estimated. This is necessary because too many parameters lead to unreliable estimates when training data is limited. In addition reducing the number of parameters reduces the computational complexity.

This problem is so serious that additional parameter limiting heuristics are used by successful recognisers.

1. The PDF from distinct states in the Markov chain may be tied, i.e. share the same parameters [48, 82, 84]. Obviously, this is only useful when they represent similar acoustics.

2. The covariance matrix is assumed to be diagonal.

3. The number of gaussians can be varied to achieve the best balance between modelling flexibility and complexity.

4. Instead of having a separate HMM for each hypothesised transcription, most systems build them from building block HMMs. These are joined with a transition from the end of one model to the start of the next. Each building block represents a fundamental speech unit such as a word, syllable or phone. The smaller the inventory of fundamental speech units the lower the number of parameters.

   The resulting embedded non-emitting states can be removed using the obvious Markov chain equivalences. To shorten the presentation we have assumed that this has been done. In an actual implementation it can sometimes be more efficient to leave these non-emitting states in place and adjust the system to compensate.

In this work results will be presented with different combinations of the above heuristics.

In order to build an effective HMM system we must make a number of choices. For descriptive purposes we shall decompose these choices into the following

1. What topology should we give our HMMs?

2. How do we efficiently find the best matching HMM?

3. How do we set the HMM parameters $\lambda = \{a_{i,j}, m_{i,k}, c_{i,k}, w_{i,k}\}$?

2.3 HMM Topology

Our HMMs were evaluated on two tasks. These were alphabet recognition on the ISOLET database [18] and e-set recognition on the CONNEX database [76]. These are standard, widely available databases on which many results have been previously reported [16–20, 40, 83].

We used 12 state HMMs to represent each letter, 10 emitting plus 2 non-emitting states. Because the final phone of the e-set members are similar the last 6 states of these models were tied. That is, state 5 of the e-set models shared the same gaussian and mixture weight, as did state 6,7,8,9 and 10. In the ISOLET database, the e-set members are {B, C, D, E, G, P, T, V, Z}, while in the CONNEX e-set database the e-set members are {B, C, D, E, G, P, T, V}.

This tying does not substantially alter the equations presented in this thesis. The modifications involved typically consist of an extra summation when calculating posterior probabilities and partial derivatives for members of a tied set.

The pre- and post-letter silence is modelled by a 3 state HMM. That is, 1 emitting and 2 non-emitting states.

These HMMs, as well as a typical composite transcription HMM, is illustrated in figure 2.1. These topologies were chosen in light of previous experience (for instance [83]), and no attempt was made to optimise them during this work.

2.4 Finding the Best Transcription

It is well known that the transcription w, that minimises the probability of error is

$$w = \arg\max_{w} P^*(w|O(u)) \tag{2.10}$$

where $P^*(w|O(u))$ the true probability of transcription w given the speech $O(u)$.

Using Bayes' rule we obtain

$$P^*(w|O(u)) = \frac{P^*(O(u)|w)P^*(w)}{P^*(O(u))} \tag{2.11}$$

This decomposition yields 3 computable factors.

$P^*(O(u)|w)$ the probability of the observation sequence $O(u)$ given the transcription w. In our recogniser this value is modelled by a synthesised HMM for w.

$P^*(w)$ the a-priori probability of the transcription w. In this thesis this is given as part of the problem specification and is kept fixed.

$P^*(O(u))$ the a-priori probability of the preprocessed speech $O(u)$. This is independent of the w and can be ignored during classification.

Since $P^*(O(u)|w)$ is unknown, we approximate it with the estimates computed by our HMM.

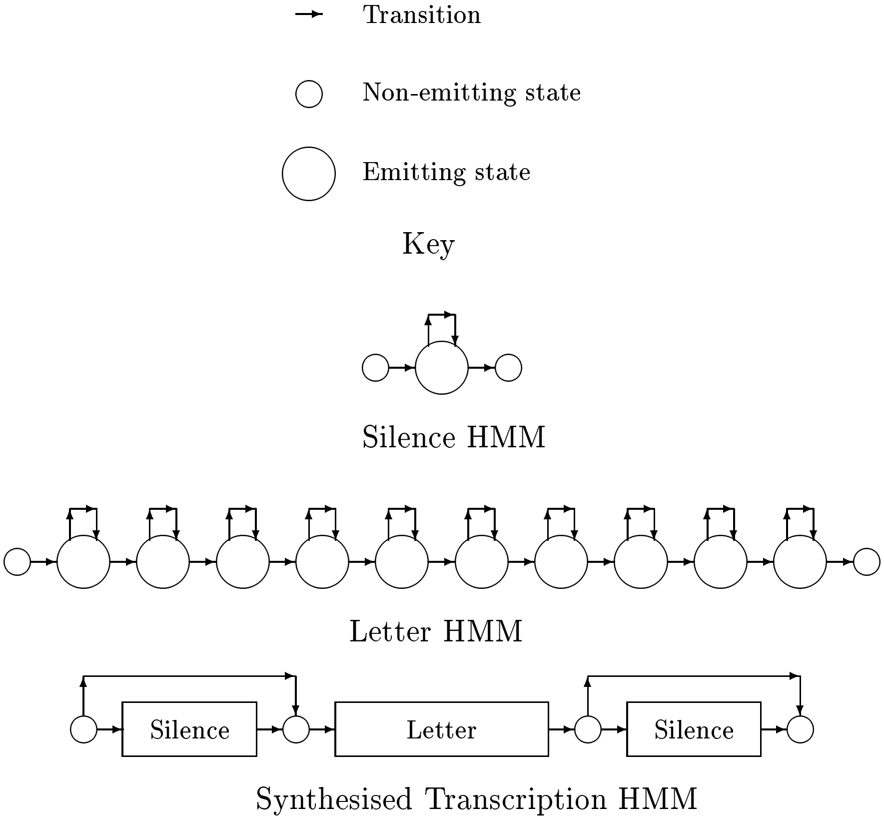

Figure 2.1: HMM topologies

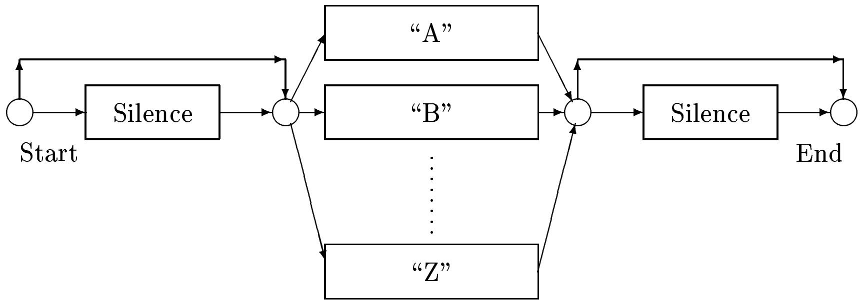

Figure 2.2: The ISOLET recognition HMM

We further approximate our HMM estimate by using the Viterbi approximation, equation 2.14.

$$P^*(O(u)|w) \approx P_\lambda(O(u)|w) \quad (2.12)$$

$$= \sum_{s \in w} \left[\prod_{t=0}^{T(u)-1} a_{s_t, s_{t+1}} \right] a_{s_{T(u)}, N} \left[\prod_{t=1}^{T(u)} b_{s_t}(O(u)_t) \right] \quad (2.13)$$

$$\approx \max_{s \in w} \left[\prod_{t=0}^{T(u)-1} a_{s_t, s_{t+1}} \right] a_{s_{T(u)}, N} \left[\prod_{t=1}^{T(u)} b_{s_t}(O(u)_t) \right] \quad (2.14)$$

where $s \in w$ indicates that s ranges over all possible full state paths in the synthesised HMM w.

Tests carried out during this work showed that the performance obtained with 2.13 and 2.14 was identical. Using this approximation allows us to obtain not only a probability estimate but also the best state path.

Direct calculation of this quantity is not feasible as the number of possible state paths is exponential in $T(u)$. Fortunately we can reduce the computation required. Firstly note that

$$\max_w \left[\max_{s \in w} P_\lambda(O(u), s|w) \right] = \max_{s \in R} P_\lambda(O(u), s|R) \quad (2.15)$$

where R is a HMM which represents all possible transcriptions in parallel. Such HMMs, suitable for recognition on the ISOLET and CONNEX e-set databases, are illustrated in figure 2.2 and 2.3. As can be seen, we have applied graph factoring techniques to reduce the number of silence models. For our work every junction in figure 2.2 and 2.3 was given equal probability. In many tasks this type of graph reduction produces huge computational savings. In our alphabet recognition tasks the savings are minor, but using R instead of w maximises the savings produced by the next two optimisations.

We can reduce the computational burden further by using the Viterbi recursion [67]. This

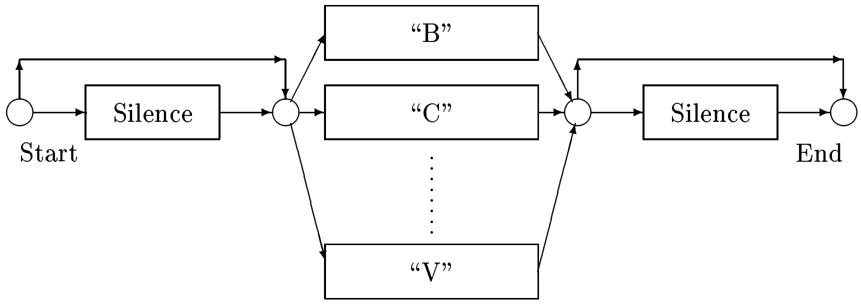

Figure 2.3: The CONNEX E-set recognition HMM

recursion on R allows us to find the most likely state sequence linearly in $T(u)$.

$$V_1(0) = 1.0 \tag{2.16}$$
$$V_j(0) = 0.0 \quad \text{for } j = 2, \ldots, N \tag{2.17}$$
$$V_1(t) = 0.0 \tag{2.18}$$
$$V_j(t) = \left[\max_i V_i(t-1)a_{i,j}\right] b_j(O(u)_t) \quad \text{for } t = 1, \ldots, T(u),\ j = 2, \ldots, N-1 \tag{2.19}$$
$$V_N(t) = \max_i V_i(t)a_{i,N} \tag{2.20}$$
$$S(T(u)) = \arg\max_i V_i(T(u))a_{i,N} \tag{2.21}$$
$$S(t) = \arg\max_i V_i(t)a_{i,S(t+1)} \quad \text{for } t = T(u)-1, \ldots, 0 \tag{2.22}$$

In the above equations $V_j(t)$ represents the joint probability, maximised over all partial state sequences of $s_t = j$ and $O(u)_1, \ldots, O(u)_t$. While $S(t)$, is the state at time t on the most likely full state path.

The final computational saving comes from beam-width pruning. $V_j(t)$ is only calculated if it is reachable from an active state at time $t-1$. Active states are states i, for which $V_i(t-1)$ is close to $\max_k V_k(t-1)$. This heuristic reduces computation with little loss in performance if the margin is chosen carefully.

2.5 Setting the Parameters

In this section we describe the most widely used learning algorithm for HMMs, maximum likelihood (ML) learning via the Baum-Welch algorithm [6, 8, 21, 50]. This section serves to emphasise our notation and provides some basic results which will be needed when we calculate derivatives and hessians. Chapter 3 examines the rationale for ML and some alternatives.

ML training involves maximisation of the probability of the speech O. This would be straightforward if we knew the speech frame/gaussian pairings of the training data. However, because it is unknown, there is no closed form solution and we must rely on iterative techniques.

The Baum-Welch algorithm uses initial estimates of the parameters $\lambda = \{a_{i,j}, m_{i,k}, c_{i,k}, w_{i,k}\}$ to obtain tentative speech frame/gaussian pairings. These are then used to obtain re-estimates $\overline{\lambda} = \{\overline{a}_{i,j}, \overline{m}_{i,k}, \overline{c}_{i,k}, \overline{w}_{i,k}\}$ having the property

$$P_{\overline{\lambda}}(O) \geq P_{\lambda}(O) \tag{2.23}$$

where $P_{\lambda}(O)$ is the probability of the U training utterances calculated with parameters λ.

$\overline{\lambda}$ then takes the place of λ and the algorithm is reapplied. This process continues until some ad-hoc convergence criterion is met. As we shall see, this algorithm also maintains the required sum to one, positivity and positive definiteness constraints 2.1, 2.2, 2.3, 2.4, 2.5.

Central to our presentation of the Baum-Welch algorithm is an auxiliary function. Its derivation is presented below.

If our training data consists of U independent identical trials then,

$$\log P_{\overline{\lambda}}(O) = \sum_{u=1}^{U} \log P_{\overline{\lambda}}(O(u)|w(u)) \tag{2.24}$$

where $w(u)$ is the synthesised HMM for the u^{th} training utterance.

Next we introduce an example speech frame/gaussian pairing m into our expression for the probability of the u^{th} training utterance. That is, m represents a connected sequence of gaussians that could have potentially produced $O(u)$.

$$\log P_{\overline{\lambda}}(O(u)|w(u)) = \log \left[\frac{P_{\overline{\lambda}}(O(u), m|w(u))}{P_{\overline{\lambda}}(m|O(u), w(u))} \right] \tag{2.25}$$

$$P_{\lambda}(m|O(u), w(u)) \log P_{\overline{\lambda}}(O(u)|w(u)) = \\ P_{\lambda}(m|O(u), w(u)) \log \left[\frac{P_{\overline{\lambda}}(O(u), m|w(u))}{P_{\overline{\lambda}}(m|O(u), w(u))} \right] \tag{2.26}$$

Adding together all instances of 2.26 we obtain,

$$\sum_{u=1}^{U} \sum_{m \in w(u)} P_{\lambda}(m|O(u), w(u)) \log P_{\overline{\lambda}}(O(u)|w(u)) = \\ \sum_{u=1}^{U} \sum_{m \in w(u)} P_{\lambda}(m|O(u), w(u)) \log \left[\frac{P_{\overline{\lambda}}(O(u), m|w(u))}{P_{\overline{\lambda}}(m|O(u), w(u))} \right] \tag{2.27}$$

where $m \in w(u)$ indicates that m ranges over all possible full speech frame/gaussian pairings in the synthesised HMM $w(u)$.

The left hand side of 2.27 reduces to $\log P_{\overline{\lambda}}(O)$. Hence,

$$\log P_{\overline{\lambda}}(O) = Q(\lambda, \overline{\lambda}) - R(\lambda, \overline{\lambda}) \tag{2.28}$$

where,

$$Q(\lambda, \overline{\lambda}) = \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \log P_{\overline{\lambda}}(O(u), m|w(u)) \tag{2.29}$$

$$R(\lambda, \overline{\lambda}) = \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \log P_{\overline{\lambda}}(m|O(u), w(u)) \tag{2.30}$$

It should be noted at this point the above derivation is valid for any $\overline{\lambda}$ and in particular $\overline{\lambda} = \lambda$.

Consider now the equality

$$\log P_{\overline{\lambda}}(O) - \log P_\lambda(O) = \left[Q(\lambda, \overline{\lambda}) - Q(\lambda, \lambda)\right] + \left[R(\lambda, \lambda) - R(\lambda, \overline{\lambda})\right] \tag{2.31}$$

But it is easily proved that

$$R(\lambda, \lambda) - R(\lambda, \overline{\lambda}) \geq 0 \tag{2.32}$$

Therefore if

$$Q(\lambda, \overline{\lambda}) - Q(\lambda, \lambda) \geq 0 \tag{2.33}$$

then

$$\log P_{\overline{\lambda}}(O) \geq \log P_\lambda(O) \tag{2.34}$$

We thus define the parameter re-estimates to be those which maximise $Q(\lambda, \overline{\lambda})$ as a function of $\overline{\lambda}$. We can further prove that the fixed point of this transformation corresponds to a critical point of the likelihood.

$$\begin{aligned}
\frac{\partial \log P_\lambda(O)}{\partial \lambda} &= \sum_{u=1}^{U} \frac{\partial \log P_\lambda(O(u)|w(u))}{\partial \lambda} && (2.35)\\
&= \sum_{u=1}^{U} \frac{1}{P_\lambda(O(u)|w(u))} \frac{\partial}{\partial \lambda} \sum_{m \in w(u)} P_\lambda(O(u), m|w(u)) && (2.36)\\
&= \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \frac{\partial \log P_\lambda(O(u), m|w(u))}{\partial \lambda} && (2.37)\\
&= \left.\frac{\partial Q(\lambda, \overline{\lambda})}{\partial \overline{\lambda}}\right|_{\overline{\lambda}=\lambda} && (2.38)
\end{aligned}$$

therefore

$$\left.\frac{\partial Q(\lambda, \overline{\lambda})}{\partial \overline{\lambda}}\right|_{\overline{\lambda}=\lambda} = 0 \tag{2.39}$$

is equivalent to

$$\frac{\partial \log P_\lambda(O)}{\partial \lambda} = 0 \tag{2.40}$$

We now proceed to calculate $\overline{a}_{i,j}$, our transition probability re-estimate.

$$Q(\lambda, \overline{\lambda}) = \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \log P_{\overline{\lambda}}(m, O(u)|w(u)) \tag{2.41}$$

but

$$\begin{aligned} \log P_{\overline{\lambda}}(m, O(u)|w(u)) = \log \overline{a}_{s(m,T(u)),N} + \sum_{t=0}^{T(u)-1} \log \overline{a}_{s(m,t),s(m,t+1)} + \\ \sum_{t=1}^{T(u)} \log \overline{w}_{s(m,t),g(m,t)} + \\ \sum_{t=1}^{T(u)} \Big[-\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left|\overline{C}_{s(m,t),g(m,t)}\right| - \\ \frac{1}{2}(O(u)_t - \overline{m}_{s(m,t),g(m,t)})^T \overline{C}^{-1}_{s(m,t),g(m,t)} (O(u)_t - \overline{m}_{s(m,t),g(m,t)}) \Big] \end{aligned} \tag{2.42}$$

where $s(m,t)$ indexes the speech frame/gaussian pairings m and returns the state at time t, and
$g(m,t)$ indexes the speech frame/gaussian pairings m and returns the gaussian at time t.

In order to impose the sum to one constraint 2.2, we introduce the Lagrange multiplier ϕ_i. As we shall see the positivity constraint 2.1, is automatically met.

Let us first examine the equality satisfied by the re-estimate for $\overline{a}_{i,j} \quad j \neq N$.

$$0 = \frac{\partial Q(\lambda, \overline{\lambda})}{\partial \overline{a}_{i,j}} - \frac{\partial}{\partial \overline{a}_{i,j}} \left[\phi_i (\sum_{j=1}^{N} \overline{a}_{i,j} - 1) \right] \tag{2.43}$$

$$= \left[\sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \sum_{t=0}^{T(u)-1} \frac{\delta_{s(m,t),i}\delta_{s(m,t+1),j}}{\overline{a}_{i,j}} \right] - \phi_i \tag{2.44}$$

$$= \left[\frac{1}{\overline{a}_{i,j}} \sum_{u=1}^{U} \sum_{t=0}^{T(u)-1} \sum_{m \in w(u)} P_\lambda(m, s_t = i, s_{t+1} = j|O(u), w(u)) \right] - \phi_i \tag{2.45}$$

$$\phi_i \overline{a}_{i,j} = \sum_{u=1}^{U} \sum_{t=0}^{T(u)-1} P_\lambda(s_t = i, s_{t+1} = j|O(u), w(u)) \tag{2.46}$$

the re-estimate for $\overline{a}_{i,N}$ satisfies

$$0 = \frac{\partial Q(\lambda,\overline{\lambda})}{\partial \overline{a}_{i,N}} - \frac{\partial}{\partial \overline{a}_{i,N}}\left[\phi_i(\sum_{j=1}^{N}\overline{a}_{i,j} - 1)\right] \tag{2.47}$$

$$= \left[\sum_{u=1}^{U}\sum_{m\in w(u)} P_\lambda(m|O(u),w(u))\frac{\delta_{s(m,T(u)),i}}{\overline{a}_{i,N}}\right] - \phi_i \tag{2.48}$$

$$= \left[\frac{1}{\overline{a}_{i,N}}\sum_{u=1}^{U}\sum_{m\in w(u)} P_\lambda(m, s_{T(u)} = i|O(u),w(u))\right] - \phi_i \tag{2.49}$$

$$\phi_i\overline{a}_{i,N} = \sum_{u=1}^{U} P_\lambda(s_{T(u)} = i|O(u),w(u)) \tag{2.50}$$

Adding 2.50 and the $N-1$ instances of 2.46, we obtain

$$\phi_i = \sum_{u=1}^{U}\sum_{t=0}^{T(u)} P_\lambda(s_t = i|O(u),w(u)) \tag{2.51}$$

therefore

$$\overline{a}_{i,j} = \frac{\sum_{u=1}^{U}\sum_{t=0}^{T(u)-1} P_\lambda(s_t = i, s_{t+1} = j|O(u),w(u))}{\sum_{u=1}^{U}\sum_{t=0}^{T(u)} P_\lambda(s_t = i|O(u),w(u))} \quad j \neq N \tag{2.52}$$

and

$$\overline{a}_{i,N} = \frac{\sum_{u=1}^{U} P_\lambda(s_{T(u)} = i|O(u),w(u))}{\sum_{u=1}^{U}\sum_{t=0}^{T(u)} P_\lambda(s_t = i|O(u),w(u))} \tag{2.53}$$

We now proceed to calculate $\overline{w}_{i,k}$, our gaussian weight re-estimate. In order to impose the sum to one constraint 2.4, we introduce the Lagrange multiplier α_i. Again the positivity constraint 2.3, is automatically met.

We require the solution of

$$0 = \frac{\partial Q(\lambda,\overline{\lambda})}{\partial \overline{w}_{i,k}} - \frac{\partial}{\partial \overline{w}_{i,k}}\left[\alpha_i(\sum_{k=1}^{K}\overline{w}_{i,k} - 1)\right] \tag{2.54}$$

using 2.42

$$= \left[\sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \sum_{t=1}^{T(u)} \frac{\delta_{s(m,t),i}\delta_{g(m,t),k}}{\overline{w}_{i,k}} \right] - \alpha_i \tag{2.55}$$

$$= \left[\frac{1}{\overline{w}_{i,k}} \sum_{u=1}^{U} \sum_{t=1}^{T(u)} \sum_{m \in w(u)} P_\lambda(m, s_t = i, g_t = k|O(u), w(u)) \right] - \alpha_i \tag{2.56}$$

$$= \frac{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))}{\overline{w}_{i,k}} - \alpha_i \tag{2.57}$$

therefore

$$\alpha_i = \sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i|O(u), w(u)) \tag{2.58}$$

and

$$\overline{w}_{i,k} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))}{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i|O(u), w(u))} \tag{2.59}$$

where g_t is the gaussian at time t.

We now derive the mean re-estimation equation. We require the solution of

$$0 = \frac{\partial Q(\lambda, \overline{\lambda})}{\partial \overline{m}_{i,k}} \tag{2.60}$$

$$= \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \sum_{t=1}^{T(u)} \overline{C}_{i,k}^{-1} (O(u)_t - \overline{m}_{i,k}) \delta_{s(m,t),i} \delta_{g(m,t),k} \tag{2.61}$$

$$= \sum_{u=1}^{U} \sum_{t=1}^{T(u)} (O(u)_t - \overline{m}_{i,k}) P_\lambda(s_t = i, g_t = k|O(u), w(u)) \tag{2.62}$$

$$\overline{m}_{i,k} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} O(u)_t P_\lambda(s_t = i, g_t = k|O(u), w(u))}{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))} \tag{2.63}$$

For the covariance re-estimate we again have to solve

$$\frac{\partial Q(\lambda, \overline{\lambda})}{\partial \overline{C}_{i,k}} = 0 \tag{2.64}$$

As we shall see below, the positive definiteness constraint 2.5, is automatically satisfied

$$
\begin{aligned}
0 = & \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \sum_{t=1}^{T(u)} \delta_{s(m,t),i} \delta_{g(m,t),k} \\
& \left[-\frac{1}{2} \frac{\partial \log |\overline{C}_{i,k}|}{\partial \overline{C}_{i,k}} - \frac{1}{2} \frac{\partial (O(u)_t - \overline{m}_{i,k})^T \overline{C}_{i,k}^{-1} \partial (O(u)_t - \overline{m}_{i,k})}{\partial \overline{C}_{i,k}} \right] \quad (2.65) \\
= & \sum_{u=1}^{U} \sum_{t=1}^{T(u)} \sum_{m \in w(u)} P_\lambda(s_t = i, g_t = k, m|O(u), w(u)) \\
& \left[-\frac{1}{2} \overline{C}_{i,k}^{-1} + \frac{1}{2} \overline{C}_{i,k}^{-1} (O(u)_t - \overline{m}_{i,k})(O(u)_t - \overline{m}_{i,k})^T \overline{C}_{i,k} \right] \quad (2.66)
\end{aligned}
$$

$$
\overline{C}_{i,k} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))(O(u)_t - \overline{m}_{i,k})(O(u)_t - \overline{m}_{i,k})^T}{\sum_{u=1}^{U} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))} \quad (2.67)
$$

Finally, below we derive the well known efficient recursions, the forward-backward recursions [6–8], for the quantities needed by our re-estimation equations. These are based on the idea of iteratively calculating the probability of being in each state for each successive t.

$$
\begin{aligned}
P_\lambda(O(u)|w(u)) &= \sum_{j=1}^{N} P_\lambda(O(u), s_t = j|w(u)) \quad (2.68) \\
&= \sum_{j=1}^{N} P_\lambda(O(u)_1, \ldots, O(u)_t, s_t = j|w(u)) \\
&\quad P_\lambda(O(u)_{t+1}, \ldots, O(u)_{T(u)}|O(u)_1, \ldots, O(u)_t, s_t = j, w(u)) \quad (2.69) \\
&= \alpha(u,t,j)\beta(u,t,j) \quad (2.70)
\end{aligned}
$$

where we have equated

$$
\begin{aligned}
\alpha(u,t,j) &= P_\lambda(O(u)_1, \ldots, O(u)_t, s_t = j|w(u)) \quad (2.71) \\
\beta(u,t,j) &= P_\lambda(O(u)_{t+1}, \ldots, O(u)_{T(u)}|O(u)_1, \ldots, O(u)_t, s_t = j, w(u)) \quad (2.72)
\end{aligned}
$$

But

$$
\alpha(u,t,j) = \sum_{i=1}^{N} P_\lambda(O(u)_1, \ldots, O(u)_t, s_t = j, s_{t-1} = i|w(u)) \quad (2.73)
$$

Given our modelling assumptions,

$$\alpha(u,t,1) = 0.0 \tag{2.74}$$

$$\alpha(u,t,j) = \sum_{i=1}^{N-1} \alpha(u,t-1,i)a_{i,j}b_j(O(u)_t) \quad 1 < j < N \tag{2.75}$$

$$\alpha(u,t,N) = \sum_{i=2}^{N-1} \alpha(u,t,i)a_{i,N} \tag{2.76}$$

With initial values

$$\alpha(u,0,1) = \begin{cases} 1.0 & \text{if } i = 1, \\ 0.0 & \text{otherwise.} \end{cases} \tag{2.77}$$

Similarly,

$$\beta(u,t,N) = 0.0 \tag{2.78}$$

$$\beta(u,t,j) = \sum_{i=2}^{N-1} a_{j,i}b_i(O(u)_{t+1})\beta(u,t+1,i) \quad 1 < j < N \tag{2.79}$$

$$\beta(u,t,1) = \sum_{i=2}^{N-1} a_{1,i}\beta(u,t,i) \tag{2.80}$$

With final values

$$\beta(u,T(u),N) = \begin{cases} 1.0 & \text{if } i = N, \\ a_{i,N} & \text{otherwise.} \end{cases} \tag{2.81}$$

$$\tag{2.82}$$

With these statistics we can easily compute various useful probabilities.

$$P_\lambda(O(u)|w(u)) = \alpha(u,T(u),N) \tag{2.83}$$

$$= \beta(u,0,1) \tag{2.84}$$

$$P_\lambda(s_t = i, s_t = 1|O(u),w(u)) = 0.0 \tag{2.85}$$

$$P_\lambda(s_t = i, s_t = N|O(u),w(u)) = \begin{cases} 0.0 & \text{if } t \neq T(u), \\ \frac{\alpha(u,T(u),i)a_{i,N}}{P_\lambda(O(u)|w(u))} & \text{otherwise.} \end{cases} \tag{2.86}$$

$$P_\lambda(s_t = i, s_{t+1} = j|O(u),w(u)) = \frac{\alpha(u,t,i)a_{i,j}b_j(O(u)_{t+1})\beta(u,t+1,j)}{P_\lambda(O(u)|w(u))} \tag{2.87}$$

$$P_\lambda(s_t = i, g_t = k|O(u),w(u)) = \frac{\sum_{j=1}^{N} \alpha(u,t-1,j)a_{j,i}w_{i,k}b_{i,k}(O(u)_t)\beta(u,t,i)}{P_\lambda(O(u)|w(u))} \tag{2.88}$$

$$P_\lambda(s_t = i|O(u),w(u)) = \frac{\alpha(u,t,i)\beta(u,t,i)}{P_\lambda(O(u)|w(u))} \tag{2.89}$$

Using equation 2.89, we can reduce the computation required in the calculation of $\beta(u,t,j)$. Equation 2.79 is only applied if $P_\lambda(s_{t+1} = i|O(u), w(u))$ is above a floor value. If this floor is smaller than the floating point accuracy of the computer, the results will be exact. In this work, we used a floor value of 10^{-6}.

Finally we can now present the Baum-Welch re-estimation formulas.

$$\overline{a}_{i,j} = \frac{\sum_{u=1}^{U}\sum_{t=0}^{T(u)-1}\alpha(u,t,i)a_{i,j}b_j(O(u)_{t+1})\beta(u,t+1,j)}{\sum_{u=1}^{U}\sum_{t=0}^{T(u)}\alpha(u,t,i)\beta(u,t,i)} \quad j \neq N \tag{2.90}$$

$$\overline{a}_{i,N} = \frac{\sum_{u=1}^{U}\alpha(u,T(u),i)a_{i,N}}{\sum_{u=1}^{U}\sum_{t=0}^{T(u)}\alpha(u,t,i)\beta(u,t,i)} \tag{2.91}$$

$$\overline{w}_{i,k} = \frac{\sum_{u=1}^{U}\sum_{t=0}^{T(u)}\sum_{j=1}^{N}\alpha(u,t-1,j)a_{j,i}w_{i,k}b_{i,k}(O(u)_t)\beta(u,t,i)}{\sum_{u=1}^{U}\sum_{t=0}^{T(u)}\alpha(u,t,i)\beta(u,t,i)} \tag{2.92}$$

$$\overline{m}_{i,k} = \frac{\sum_{u=1}^{U}\sum_{t=0}^{T(u)}\sum_{j=1}^{N}O(u)_t\alpha(u,t-1,j)a_{j,i}w_{i,k}b_{i,k}(O(u)_t)\beta(u,t,i)}{\sum_{u=1}^{U}\sum_{t=0}^{T(u)}\sum_{j=1}^{N}\alpha(u,t-1,j)a_{j,i}w_{i,k}b_{i,k}(O(u)_t)\beta(u,t,i)} \tag{2.93}$$

$$\overline{C}_{i,k} = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T(u)}\sum_{j=1}^{N}(O(u)_t-\overline{m}_{i,k})(O(u)_t-\overline{m}_{i,k})^T\alpha(u,t-1,j)a_{j,i}w_{i,k}b_{i,k}(O(u)_t)\beta(u,t,i)}{\sum_{u=1}^{U}\sum_{t=1}^{T(u)}\sum_{j=1}^{N}\alpha(u,t-1,j)a_{j,i}w_{i,k}b_{i,k}(O(u)_t)\beta(u,t,i)} \tag{2.94}$$

2.6 Summary

This chapter has reviewed the theory behind our baseline modelling approach. Firstly we described our modelling technique — HMMs. We then showed how we can use our HMMs with the Viterbi algorithm to classify speech. Finally, we presented a simple derivation of the Baum-Welch algorithm for ML learning.

These techniques are the ones used by almost all successful speech recognition systems. They provide a baseline against which we shall evaluate our new methods. In chapter 3 we present alternatives to the ML objective function. In addition, we show that the Viterbi algorithm for decoding remains valid. These alternative objective functions require new training algorithms and these are described in chapter 4.

Chapter 3

Objective Functions

This chapter reviews in depth the theoretical justifications for various objective functions. I have chosen to be selective in my coverage. The objective functions described are considered by me to be either effective, important due to their widespread use or of theoretical value.

Firstly we review the theory behind ML estimators. We will show that they can be justified without making any incorrect assumptions and so are optimal on many occasions. We then describe three examples of ML estimators, the traditionally but confusingly named ML, maximum mutual information (MMI) and frame discrimination (FD). Finally we shall describe one non-ML estimator that is currently very popular, classification figure of merit (CFM).

3.1 Properties of Maximum Likelihood Estimators

In this section we will discuss ML estimators abstractly. That is, we will represent the event whose probability we are maximising by the symbol E. For a training set consisting of U examples E represents the joint event $E(1), \ldots, E(U)$, where for example,

$$E(u) = \begin{cases} \text{for speech literature ML, the event } O(u) \text{ given } w(u) \text{ in the domain of all } O(u), \\ \text{for MMI, the event } w(u) \text{ given } O(u) \text{ in the domain of } w(u) \in R, \\ \text{for FD, the event } s_1, \ldots, s_{T(u)} \text{ given } O(u) \text{ in the domain of } s_1, \ldots, s_{T(u)} \in N. \end{cases} \tag{3.1}$$

In this way we can prove the properties shared by all ML estimators.

Past studies have emphasised the role of ML estimators in finding confidence intervals for correct models. In this context it is known that the method has extremely good justification [66, 81]. For instance,

1. the ML estimator is a function of sufficient statistic t, if a sufficient statistic exists. A statistic is said to be sufficient for a parameter λ, if the conditional distribution of the observations given the value of the statistic is independent of λ. That is,

$$\lambda_{\mathrm{ML}} = \arg\max_{\lambda} f(t) \tag{3.2}$$

where,

$$P_{\lambda_i}(E|t) = P_{\lambda_j}(E|t) \quad \forall i,j \tag{3.3}$$

Equation 3.3 shows us that the sufficient statistic contains as much information about the parameter λ as the original observations. And equation 3.2 shows us that ML is a function of only this information..

2. In the large sample case, the ML estimator distribution becomes normal and unbiased with a covariance matrix reaching the Cramer-Rao lower bound for the covariance of an unbiased estimator [81].

We shall now prove the sufficiency of ML. Let t, a function of the training data, be an estimator of λ. Then the following identity is true.

$$P_\lambda(E) = P_\lambda(E,t) \tag{3.4}$$
$$= P_\lambda(t)P_\lambda(E|t) \tag{3.5}$$

if t is sufficient for λ then,

$$\frac{\partial P_\lambda(E|t)}{\partial \lambda} = 0 \tag{3.6}$$

In other words, $P_\lambda(E|t)$ is independent of λ and can be written as $P(E|t)$. This gives us,

$$\frac{\partial P_\lambda(E)}{\partial \lambda} = \frac{\partial P_\lambda(t)}{\partial \lambda} P(E|t) \tag{3.7}$$

Which tells us that the ML estimator, which is the solution to,

$$\frac{\partial P_\lambda(E)}{\partial \lambda} = 0 \tag{3.8}$$

is equivalent to,

$$\frac{\partial P_\lambda(t)}{\partial \lambda} = 0 \tag{3.9}$$

That is, the ML estimator is a function of t.

To derive the asymptotic distribution of the ML estimate we need to prove some properties of the distribution of $\frac{\partial \log P_\lambda(E)}{\partial \lambda}$. For the purposes of our proofs we shall assume those regularity conditions necessary to make our manipulations valid.

1. The expected value of $\frac{\partial \log P_\lambda(E)}{\partial \lambda}$ is zero.

2. Its covariance matrix is $-\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)$.

3. In the large sample case it is approximately normally distributed.

We can see that $\frac{\partial \log P_\lambda(E)}{\partial \lambda}$ is normally distributed since it is the sum of u independent variables

$$\frac{\partial \log P_\lambda(E)}{\partial \lambda} = \sum_{u=1}^{U} \frac{\partial \log P_\lambda(E(u))}{\partial \lambda} \tag{3.10}$$

hence we may invoke the central limit theorem.

We now prove 1,

$$\begin{aligned}
\mathcal{E}\left(\frac{\partial \log P_\lambda(E)}{\partial \lambda}\right) &= \int \cdots \int \frac{\partial \log P_\lambda(E)}{\partial \lambda} P_\lambda(E) dO(1) \cdots dO(U) && (3.11)\\
&= \int \cdots \int \frac{\partial P_\lambda(E)}{\partial \lambda} dO(1) \cdots dO(U) && (3.12)\\
&= \frac{\partial}{\partial \lambda} \int \cdots \int P_\lambda(E) dO(1) \cdots dO(U) && (3.13)\\
&= 0 && (3.14)
\end{aligned}$$

To prove 2, we have by definition,

$$\begin{aligned}
\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2} &= \frac{\partial}{\partial \lambda}\left(\frac{1}{P_\lambda(E)} \frac{\partial P_\lambda(E)}{\partial \lambda}\right) && (3.15)\\
&= -\frac{\partial \log P_\lambda(E)}{\partial \lambda} \frac{\partial \log P_\lambda(E)}{\partial \lambda}^T + \frac{1}{P_\lambda(E)} \frac{\partial^2 P_\lambda(E)}{\partial \lambda^2} && (3.16)
\end{aligned}$$

using 3.14,

$$\begin{aligned}
\mathrm{Cov}\left(\frac{\partial \log P_\lambda(E)}{\partial \lambda}\right) &= \mathcal{E}\left(\frac{\partial \log P_\lambda(E)}{\partial \lambda} \frac{\partial \log P_\lambda(E)}{\partial \lambda}^T\right) && (3.17)\\
&= \mathcal{E}\left(-\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2} + \frac{1}{P_\lambda(E)} \frac{\partial^2 P_\lambda(E)}{\partial \lambda^2}\right) && (3.18)\\
&= -\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right) + \int \cdots \int \frac{\partial^2 P_\lambda(E)}{\partial \lambda^2} dO(1) \cdots dO(U) && (3.19)\\
&= -\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right) + \frac{\partial^2}{\partial \lambda^2} \int \cdots \int P_\lambda(E) dO(1) \cdots dO(U) && (3.20)\\
&= -\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right) && (3.21)
\end{aligned}$$

With these preliminary results, we can now find the sampling distribution of the ML estimate of λ, λ_{ML}. $\frac{\partial \log P_\lambda(E)}{\partial \lambda}$ evaluated at λ_{ML} rather than at the true value λ^*, is by definition the zero vector. By Taylor's theorem this quantity is approximately equal to

$$\begin{aligned}
\left.\frac{\partial \log P_\lambda(E)}{\partial \lambda}\right|_{\lambda=\lambda_{\mathrm{ML}}} &= 0 && (3.22)\\
&\approx \left.\frac{\partial \log P_\lambda(E)}{\partial \lambda}\right|_{\lambda=\lambda^*} + \left.\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right|_{\lambda=\lambda^*} (\lambda_{\mathrm{ML}} - \lambda^*) && (3.23)
\end{aligned}$$

therefore,

$$\lambda_{\mathrm{ML}} - \lambda^* \approx - \left(\left. \frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2} \right|_{\lambda=\lambda^*} \right)^{-1} \left. \frac{\partial \log P_\lambda(E)}{\partial \lambda} \right|_{\lambda=\lambda^*} \tag{3.24}$$

In the large sample case, we can replace the observed value of $\left. \frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2} \right|_{\lambda=\lambda^*}$ with its expected value.

From the sampling distribution of $\frac{\partial \log P_\lambda(E)}{\partial \lambda}$ it follows that the expected value of λ_{ML} is λ^*, its covariance matrix is $-\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)^{-1}$ and that it tends to be normally distributed in large samples.

We expect any estimator to converge onto true parameters when we have an infinite amount of data. So asymptotic unbiasedness is a desirable property of an estimator. We shall now prove that no unbiased estimator has a lower dispersion than ML.

Now consider the covariance matrix of $\begin{bmatrix} t \\ \frac{\partial \log P_\lambda(E)}{\partial \lambda} \end{bmatrix}$, where t is an unbiased estimator of λ. This is given by,

$$\begin{bmatrix} \mathrm{Cov}(t) & \mathrm{Cov}(t, \frac{\partial \log P_\lambda(E)}{\partial \lambda}) \\ \mathrm{Cov}(t, \frac{\partial \log P_\lambda(E)}{\partial \lambda}) & \mathrm{Cov}(\frac{\partial \log P_\lambda(E)}{\partial \lambda}) \end{bmatrix} = \begin{bmatrix} \mathrm{Cov}(t) & \mathcal{I} \\ \mathcal{I} & -\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right) \end{bmatrix} \tag{3.25}$$

where,
$\mathcal{I}$ represents an identity matrix,
$\mathrm{Cov}(t)$ represents the variance of t, and
$\mathrm{Cov}(t, \frac{\partial \log P_\lambda(E)}{\partial \lambda})$ represents the covariance of t and $\frac{\partial \log P_\lambda(E)}{\partial \lambda}$.

Since covariance matrices are positive semi-definite, it follows that

$$\begin{bmatrix} \mathcal{I} & \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)^{-1} \end{bmatrix} \begin{bmatrix} \mathrm{Cov}(t) & \mathcal{I} \\ \mathcal{I} & -\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right) \end{bmatrix} \begin{bmatrix} \mathcal{I} \\ \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)^{-1} \end{bmatrix} = \mathrm{Cov}(t) + \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)^{-1} \tag{3.26}$$

is also semi-positive definite. In order to better understand this result, consider a linear combination of the estimated parameters ct, where c are the weighting factors. Because t is an unbiased estimator of λ, ct is an unbiased estimator of $c\lambda$. Its variance is therefore $c^T \mathrm{Cov}(t) c$. But using the above result we obtain,

$$c^T \left(\mathrm{Cov}(t) + \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)^{-1} \right) c \geq 0 \tag{3.27}$$

which gives,

$$\mathrm{Cov}(ct) \geq -c^T \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(E)}{\partial \lambda^2}\right)^{-1} c \tag{3.28}$$

That is, the asymptotic dispersion of a ML estimate cannot be bettered by any unbiased estimator.

For fields in which the model is the true generator of the data sufficiency and efficiency is a powerful justification for the ML method. Sufficiency allows us to reduce the memory requirements of our training procedure by storing t instead of the raw training data. In addition, efficiency gives us the peace of mind that comes from the knowledge, that the ML estimator has no larger variance than a very large class of good estimators — the class of unbiased estimators.

However, a HMM is not the true generator for speech, in fact the true generator is unknown. Clearly the above results cannot be applied to the speech recognition field. Because of this, it has been stated by some speech researchers that ML is suboptimal. But the reasoning behind this statement is flawed.

We assume model correctness in order to prove sufficiency and efficiency. But this does not allow us to say that ML is suboptimal when the model is incorrect. On the contrary, in the absence of further information about the properties of ML estimators, we should remain biased towards them.

In fact we can extend our analysis to provide an intuitive justification for the small sample inexact model case. Firstly, it is obvious that ML is invariant to reparameterisation. If λ_{ML} is the ML estimate of λ, then $g(\lambda_{\mathrm{ML}})$ is the ML estimate of $g(\lambda)$. This property ensures that any front-end reparameterisation that is within the representational abilities of the model has no effect.

In addition we can write the ML parameters as,

$$\lambda_{\mathrm{ML}} = \arg\max_{\lambda} \sum_{u=1}^{U} \log P_{\lambda}(E(u)) \tag{3.29}$$

$$= \arg\max_{\lambda} \sum_{u=1}^{U} [\log P_{\lambda}(E(u)) - \log P^{*}(E(u))] \tag{3.30}$$

$$= \arg\min_{\lambda} \frac{1}{U} \sum_{u=1}^{U} \log \frac{P^{*}(E(u))}{P_{\lambda}(E(u))} \tag{3.31}$$

To understand 3.31, let us take the limit for $U \to \infty$.

$$\lambda_{\mathrm{ML}} = \arg\min_{\lambda} \mathcal{E} \left[\log \frac{P^{*}(E(u))}{P_{\lambda}(E(u))} \right] \tag{3.32}$$

the quantity being minimised is the Kullback-Leibler distance between the probability estimates supplied by HMM model $P_{\lambda}(E(u))$, and the unknown true source distribution, $P^{*}(E(u))$. Therefore the ML method seeks to minimise the sampled Kullback-Leibler distance between $P_{\lambda}(E(u))$ and $P^{*}(E(u))$. The consistency of ML is easy to understand. This distance is at its minimum of zero when $P_{\lambda}(E(u))$ matches $P^{*}(E(u))$.

One important and explicit feature of the ML method is rarely pondered upon. Measured with a ML objective function the value of a model depends only on the model's estimate of $P^{*}(E(u))$. That is, the models value does not depend upon the model's estimate of the incorrect classes. This means that only $P_{\lambda}(E(u))$ needs to be calculated and adjusted during training.

The model's estimate of other probabilities are irrelevant and can be ignored. This often results in a computational saving during the training process.

We can show that ML is unique in having this property of "locality" as it is known [1, 25]. For a training set consisting of $E(u), u = 1, \ldots, U$ where $E(u) \in \{1, \ldots, n\}$ we require the objective function $F(.)$ to have its maximum when our probability estimates $P_\lambda(E(u))$ are close to the true probabilities $P^*(E(u))$. If $E(u)$ occurs, by locality our model will be valued at $F(P_\lambda(E(u)))$. That is,

$$\sum_{u=1}^{U} F(P^*(E(u)) \approx \max_{P_\lambda} \sum_{u=1}^{U} F(P_\lambda(E(u))) \tag{3.33}$$

In the limit as $U \to \infty$ we require 3.33 to be an equality.

$$\sum_{i=1}^{n} P^*(i) F(P^*(i)) = \max_{P_\lambda} \sum_{i=1}^{n} P^*(i) F(P_\lambda(i)) \tag{3.34}$$

In order to prove that ML is unique in displaying the property of locality we have to show that the only solution to equation 3.34 occurs when $F(.) = \log(.)$. In order to find the maximum of 3.34 under the probabilistic constraints for $P_\lambda(E(u))$, we introduce the auxiliary variable θ. Differentiating the lagrangian we obtain,

$$P^*(j) \left. \frac{\partial F(P_\lambda(j))}{\partial P_\lambda(j)} \right|_{P_\lambda(j) = P^*(j)} - \theta = 0 \tag{3.35}$$

$$\frac{\partial F(P^*(j))}{\partial P^*(j)} = \frac{\theta}{P^*(j)} \tag{3.36}$$

Therefore,

$$F(P^*(j)) = \theta \log P^*(j) + C \tag{3.37}$$

with

$$\theta > 0 \tag{3.38}$$

and C is an arbitrary constant of integration.

Summarising, we have shown that the large sample exact model optimality and the computationally efficient distribution approximation outside this region can be viewed as heuristic justification of ML.

3.2 Maximum Likelihood

ML is the commonest objective criteria used for HMM training. Its almost complete dominance is the result of empirically good performance and the availability of the fast, globally convergent Baum-Welch training algorithm described in section 2.5.

The reader should note that the term ML in mathematical literature refers to a method which maximises a model likelihood. In speech literature it has been taken to mean L given by,

$$L = \sum_{1}^{U} P_\lambda(O(u)|w(u)) \tag{3.39}$$

That is, in the notation of the section 3.1, E is the event $O(u)$ given $w(u)$. This inappropriate nomenclature coupled with the incorrect assertion that ML requires model correctness assumption has lead to the widely accepted view that the use of ML cannot be justified. In fact many of the alternatives that have been proposed are traditional ML estimators. They differ from ML in that the synthesised transcription HMMs are used as a component of a larger model. For instance, in MMI training we directly optimise the Bayes' classification statistic. It is an accident of history that MMI was not named ML. The name ML was already taken when MMI was invented. The writer hopes that he has dispelled the notion that L is fundamentally flawed and proved that L is well founded.

Does L exhibit any bad attributes? In the limited sample case with our models a degenerate condition can occur. If the mean of any gaussian coincides with any observation vector and the corresponding variance tends to zero, L will increase without limit [58]. The other parameters become irrelevant to L. This leads to poor generalisation on novel test data.

Although we wish to increase the likelihood we cannot allow it to achieve its degenerate maximum. In other fields this insufficient data problem is traditionally solved with the use of Bayes' smoothing techniques. However, in this work, as is most commonly the case in the speech recognition field, we rely on the joint effect of initial parameter values and the suboptimal optimisation techniques to avoid this problem. Our initial point is chosen so that many acoustic observations map onto each Gaussian. Then practical experience has shown us that hill climbing procedures do not fall into these pathological regions of parameter space. Experience has also shown us that L produces accurate systems.

In the rest of this thesis we search on other bad attributes and seek to answer the question, "can we do better?"

3.3 Maximum Mutual Information

During MMI training our objective function is

$$I = \sum_{u=1}^{U} \log P_\lambda(w(u)|O(u)) \tag{3.40}$$

This expression can easily be derived from the expression for empirical mutual information MI, hence its name.

$$\mathrm{MI} = \sum_{u=1}^{U} \log\left[\frac{P_\lambda(O(u), w(u))}{P^*(w(u))P_\lambda(O(u))}\right] \tag{3.41}$$

$$= \sum_{u=1}^{U} \left[\log P_\lambda(w(u)|O(u)) - \log P^*(w(u))\right] \tag{3.42}$$

$$= I - \sum_{u=1}^{U} \log P^*(w(u)) \tag{3.43}$$

During the maximisation of MI over the HMM parameters λ the second term in 3.43 can be ignored.

The reader may now be wondering why MI is worthy of study. There are a number of reasons. Comparing I with the Bayes' classification criterion 2.10 we see that they use the same statistic. This ensures optimality on the test set as the training set size becomes large.

Comparing L to MMI, we see that L maximises the conditional probability of the observation given the model, while in MMI we maximise the conditional probability of the transcription. If $P_\lambda(O(u)|w(u))$ is correct, both have the optimal properties of a ML estimator. In particular, both produce estimates which are asymptotically unbiased, normal and, given the limitations of their assumed source distributions, have the minimum dispersion possible. But as we shall show next, the asymptotic dispersion of the ML estimate is lower than the asymptotic dispersion of the MMI estimate.

$$\log P_\lambda(O(u)|w(u)) + \log P^*(w(u)) = \log P_\lambda(w(u)|O(u)) + \log P_\lambda(O(u)) \tag{3.44}$$

$$\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial\lambda^2}\right) = \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial\lambda^2}\right) + \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u))}{\partial\lambda^2}\right) \tag{3.45}$$

For any arbitrary constant vector c,

$$c^T\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial\lambda^2}\right)c - c^T\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial\lambda^2}\right)c = c^T\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u))}{\partial\lambda^2}\right)c \tag{3.46}$$

Using 3.17 and 3.21 we obtain,

$$\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u))}{\partial\lambda^2}\right) = -\mathcal{E}\left(\frac{\partial \log P_\lambda(O(u))}{\partial\lambda}\frac{\partial \log P_\lambda(O(u))}{\partial\lambda}^T\right) \tag{3.47}$$

Using this identity with 3.46 we obtain,

$$c^T\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial\lambda^2}\right)c - c^T\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial\lambda^2}\right)c \geq 0 \tag{3.48}$$

This is a strict inequality if $\mathcal{E}\left(\frac{\partial \log P_\lambda(O(u))}{\partial \lambda}\frac{\partial \log P_\lambda(O(u))}{\partial \lambda}^T\right)$ is positive definite. That is, if the marginal distribution that is ignored by MMI $P_\lambda(O(u))$, provides some information about λ. From equation 3.48 we can obtain,

$$c^T \left[\ \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial \lambda^2}\right)^{-1} \quad \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial \lambda^2}\right)^{-1}\ \right] \left[\begin{array}{ll} \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial \lambda^2}\right) & 0 \\ 0 & -\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial \lambda^2}\right) \end{array}\right] \left[\begin{array}{l} \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial \lambda^2}\right)^{-1} \\ \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial \lambda^2}\right)^{-1} \end{array}\right] c \geq 0 \tag{3.49}$$

Which yields,

$$-c^T \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial \lambda^2}\right)^{-1} c \leq -c^T \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(w(u)|O(u))}{\partial \lambda^2}\right)^{-1} c \tag{3.50}$$

That is, the dispersion of the ML estimate is less than or equal to that of the MMI estimate with a strict inequality only if $P_\lambda(O(u))$ provides no information about λ.

Notice that $P_\lambda(O(u))$ can be calculated from our ML models but not from the MMI models. MMI places less demands on the representational abilities of our models than ML. It could be the case for instance, that $P_\lambda(w(u)|O(u))$ could be the correct model for some λ while $P_\lambda(O(u)|w(u))$ is not correct for any λ. Then asymptotically MMI should be favoured. In the incorrect model case, because we are not approximating $P_\lambda(O(u))$, the available parameters could be more profitably used to model the conditional probabilities needed for Bayes' decoding rule. If this behaviour extends to an independent test set then we should achieve an improved recognition rate. We therefore have some hope of finding that MMI is better than ML.

The vital difference between ML and MMI is that the two methods model different source distributions. MMI minimises the empirical sampled distance between the probability estimate supplied by our model $P_\lambda(w(u)|O(u))$ and the true posterior probability of the word given the acoustic observations $P^*(w(u)|O(u))$. Whereas, ML minimises the empirical sampled distance between $P_\lambda(O(u)|w(u))$ and $P^*(O(u)|w(u))$.

This point is the key to the motivation behind this thesis. So it is worth emphasising. In ML training our HMMs are trained to model $P^*(O(u)|w(u))$. During classification we use our estimate $P_\lambda(O(u)|w)$ to indirectly compute the classification statistic $P_\lambda(w|O(u))$ via the identity

$$P_\lambda(w|O(u)) = \frac{P_\lambda(O(u)|w)P^*(w)}{\sum_w P_\lambda(O(u)|w)P^*(w)} \tag{3.51}$$

To improve performance for problem transcriptions, we must explicitly allocate resources. For instance we could increase the number of gaussians in the correct synthesised HMM or in the

HMMs confused with it. During MMI training we directly model $P^*(w(u)|O(u))$. From 3.51, we can see that the parameters of all the HMMs are adjusted simultaneously. The MMI training process effectively allocates resources according to acoustic difficulty. MMI therefore uses the same parameters more efficiently.

To recap the MMI objective function is,

$$\begin{aligned} I &= \sum_{u=1}^{U} \log P_\lambda(w(u)|O(u)) && (3.52) \\ &= \sum_{u=1}^{U} \log P_\lambda(w(u), O(u)) - \sum_{u=1}^{U} \log P_\lambda(O(u)) && (3.53) \\ &= \sum_{u=1}^{U} [\log P_\lambda(O(u)|w(u)) + \log P^*(w(u))] - \sum_{u=1}^{U} \log \sum_{\forall w} P_\lambda(w, O(u)) && (3.54) \end{aligned}$$

We can see the first term is identical to the quantity used in ML training. It can therefore be calculated in the same manner. The second term is obtained by applying the Baum-Welch algorithm to the recognition HMM R, that is,

$$\sum_{\forall w} P_\lambda(w, O(u)) = P_\lambda(O(u)|R) \tag{3.55}$$

In most tasks this is the most computationally expensive step in MMI training. Calculation of the MMI objective function requires a recognition pass on each of the training utterances.

Previously we mentioned that during classification with a MMI HMM, equation 3.51 has to be calculated for each alternative w and the w that yields the maximum is chosen. The reader should note that the denominator is independent of w and so can be ignored. In addition when we calculate the numerator, by using the Viterbi approximation we reduce the recognition process to the standard ML approach.

3.4 Frame Discrimination

This section presents a class of novel objective functions. These objective functions, as we shall explain, have the potential for better training set performance than ML and better generalisation than MMI.

To recap, the MMI objective function is

$$\begin{aligned} I &= \sum_{u=1}^{U} \log P_\lambda(w(u)|O(u)) && (3.56) \\ &= \sum_{u=1}^{U} [\log P_\lambda(O(u)|w(u)) + \log P^*(w(u)) - \log P_\lambda(O(u)|R)] && (3.57) \end{aligned}$$

Looking at this expression, experienced practitioners can identify a possible source of problems. The two terms $P_\lambda(O(u)|w(u))$ and $P_\lambda(O(u)|R)$ are each very likely to be dominated by very few paths. If these paths are the same, that is they both belong to $w(u)$, we achieve a good rating for $O(u)$. If these paths are different, then our aim is to make the correct path more likely. If we restrict our attention to a specimen observation $O(u)_t$, then we may equivalently say, our aim is to separate the correct and incorrect sets of state distributions associated with $O(u)_t$. The match of other states to $O(u)_t$ is irrelevant. To make matters worse, confusions at time t can also be removed by improvements at times other than t due to state sequence coupling. There is a danger that the training set confusions that do exist, will not be learned.

Consider now an independent test set. Our model's rating on this test set depends on the training set's coverage of confusible states. The smaller the training set or equivalently the greater the HMMs flexibility the worse our test set performance will be.

If the modelling flexibility of our HMM is large or our training data is limited, it may be profitable to increase the number of confused states. Of course, we do not know which states are confused on novel data. If we did we would not be in this situation. The best we can hope for, when we introduce extra confusions is that the invalid confusions do not swamp out the valid ones.

One possible approach to introducing extra confusions is to replace the recognition model R, with another HMM N, especially constructed to increase the number of confusions. That is the new HMM N should allow a superset of the state sequences that is allowed by R. Substituting into 3.57 we obtain our FD objective function D.

$$D = \sum_{u=1}^{U} [\log P_\lambda(O(u)|w(u)) + \log P^*(w(u)) - \log P_\lambda(O(u)|N)] \tag{3.58}$$

As with the original MMI calculating the value of the above requires two forward-backward passes per training example. The first on the synthesised transcription HMM $w(u)$, and the second on the confusion HMM N. Because of the greater number of allowed alignments, 3.58 will underestimate I. Equivalently, when training with 3.57, the difference between the transcription HMM and recognition HMM probabilities will be larger. This in almost all cases will mean that models trained with 3.57 will have higher training set accuracy.

Balancing this we have already mentioned why 3.58, may generalise better. In addition in many tasks though not ours, handling the language model is a major share of computational complexity. Therefore in many tasks, the second term in 3.57 is more expensive to compute than the third term in 3.58.

This objective function is similar to the FD objective function used by many HMM-neural network (NN) hybrids [26, 72]. We can manipulate 3.58 in order to illustrate this more clearly.

$$D = \sum_{u=1}^{U} \left[\log \frac{P_\lambda(O(u)|w(u))}{P_\lambda(O(u)|N)} + \log P^*(w(u))\right] \tag{3.59}$$

$$= \sum_{u=1}^{U} \left[\left(\log \frac{\sum_{s \in w(u)} P_\lambda(s|w(u)) P_\lambda(O(u)|s, w(u))}{P_\lambda(O(u)|N)}\right) + \log P^*(w(u))\right] \tag{3.60}$$

$$= \sum_{u=1}^{U} \left[\left(\log \sum_{s \in w(u)} \frac{P_\lambda(s|w(u)) P_\lambda(O(u)|s, w(u)) P_\lambda(s|N)}{P_\lambda(s|N) P_\lambda(O(u)|N)}\right) + \log P^*(w(u))\right] \tag{3.61}$$

If N is not a null Markov model (i.e. $P_\lambda(O(u)|s,N)$ is not constant—this would reduce the FD objective function to ML), and N is constructed so that it allows a superset of the state paths from R, then,

$$P_\lambda(O(u)|s,w(u)) = P_\lambda(O(u)|s,N) \quad \forall s \in w(u) \tag{3.62}$$

This allows us to simplify to,

$$D = \sum_{u=1}^{U} \left[\left(\log \sum_{s \in w(u)} \frac{P_\lambda(s|w(u)) P_\lambda(s|N,O(u))}{P_\lambda(s|N)} \right) + \log P^*(w(u)) \right] \tag{3.63}$$

The $P_\lambda(s|N,O(u))$ term in equation 3.63 is the objective function used by many HMM-NN hybrids [11, 12, 38, 56, 57, 68, 69, 71]. The $\frac{P_\lambda(s|w(u))}{P_\lambda(s|N)}$ term represents the static to transcription prior compensation needed when hybrids are used to align data. Unlike HMM-NN hybrids our training procedures do not require separate alignment and fitting stages.

During recognition we calculate equation 3.58 for each alternative w and pick the w which produces the maximum. Because the $P_\lambda(O(u)|N)$ is independent of w it can be ignored. In addition if we make the Viterbi assumption, the recognition process reduces to the standard ML one.

It is easy to show that like MMI, the FD estimator has a variance larger than the ML estimator. Letting D_u be the value of the FD objective function on the u^{th} training example, we have the relation,

$$D_u = \log P_\lambda(O(u)|w(u)) + \log P^*(w(u)) - \log P_\lambda(O(u)|N) \tag{3.64}$$

Which yields by some simple manipulation,

$$\mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial \lambda^2}\right) = \mathcal{E}\left(\frac{\partial^2 D_u}{\partial \lambda^2}\right) + \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|N)}{\partial \lambda^2}\right) \tag{3.65}$$

Using 3.65 and the same reasoning used in equations 3.46–3.50 we obtain,

$$-c^T \mathcal{E}\left(\frac{\partial^2 \log P_\lambda(O(u)|w(u))}{\partial \lambda^2}\right)^{-1} c \leq -c^T \mathcal{E}\left(\frac{\partial^2 D_u}{\partial \lambda^2}\right)^{-1} c \tag{3.66}$$

That is, the dispersion of the ML estimate is less than or equal to that of the FD estimate. This is a strict inequality if $\mathcal{E}\left(\frac{\partial \log P_\lambda(O(u)|N)}{\partial \lambda} \frac{\partial \log P_\lambda(O(u)|N)}{\partial \lambda}^T\right)$ is positive definite. That is, if $P_\lambda(O(u)|N)$, provides some information about λ.[1]

In this dissertation, we report the results of experiments using a zero memory Markov chain for N. That is,

$$\begin{aligned} P_\lambda(O(u)|N) &= \prod_{t=1}^{T(u)} \sum_{i=2}^{N-1} P_\lambda(O(u)_t|q_i) P(q_i|N) && (3.67) \\ &= \prod_{t=1}^{T(u)} \sum_{i=2}^{N-1} b_i(O(u)_t|q_i) P(q_i|N) && (3.68) \end{aligned}$$

[1]The author has not yet derived the relative efficiencies of MMI and FD.

For the experiments in this thesis we set the state priors $P(q_i|N)$, to a constant independent of i.

Test experiments showed that with limited parameter models there was a slight performance improvement when utilising N with more memory. But these were not statistically significant on our tasks.

Utilising this form of N in 3.63 we obtain,

$$D = \sum_{u=1}^{U} \left[\left(\log \sum_{s \in w(u)} \frac{P_\lambda(s|w(u))}{P(s|N)} \prod_{t=1}^{T(u)} P_\lambda(s_t|O(u)_t, N) \right) + \log P^*(w(u)) \right] \tag{3.69}$$

Comparing FD to ML we can see that ML state PDFs approximate $P(O(u)_t|s_t)$, instead D adjusts the state PDFs to approximate $P(s_t|O(u)_t)$. Since the approximation of $P(s_t|O(u)_t)$ does not model $P(O(u)_t)$ and $P(O(u)_t)$ is not used in classification, D could profitably use the extra freedom to improve the training set performance. We also expect that zero memory FD will generalise to an independent test set as well as ML. In both cases we assume a worst case confusion environment (ML makes this assumption by modelling each $P(O(u)_t|s_t)$ separately, which leads to a worst case confusion environment estimate of $P(s_t|O(u)_t)$). On the other hand, as we shall see in chapter 4, all discriminative objective functions are difficult to optimise. The theoretical advantage enjoyed by FD of less "regressional stress", could be negated by the lack of the ability to find better regressions.

3.5 Differential Learning

Instead of approximating probability distributions, differential learning attempts to maximise training set accuracy. The training set accuracy is approximated by a continuous differentiable function. This allows the use of efficient gradient based optimisation techniques.

One approximation, classification figure of merit (CFM), was suggested in [32] and is given by,

$$D = \sum_{u=1}^{U} \sigma\left[\Delta(u), k\right] \tag{3.70}$$

where,

$$\Delta(u) = P_\lambda(w(u)|O(u)) - P_\lambda(v(u)|O(u)) \tag{3.71}$$

and $v(u)$ is the transcription, excluding $w(u)$ which has maximal probability on utterance $O(u)$ when calculated with parameters λ. In other words, if $O(u)$ is correctly recognised, $v(u)$ is the second ranked transcription, otherwise it is the top ranked transcription.
σ is a continuous, differentiable approximation to a step function and,
k is a parameter which controls the steepness of $\sigma\left[\Delta(u), k\right]$ when $\Delta(u) = 0$.

One suitable form for $\sigma\left[\Delta(u), k\right]$ is

$$\sigma\left[\Delta(u), k\right] = \frac{1}{1 + e^{-k\Delta(u)}} \tag{3.72}$$

It can be easily verified that as $k \to \infty$, D becomes a closer estimate of training set accuracy. On the downside, its graph will develop vertical "cliffs" and horizontal "plateaus". Both features will make it hard to optimise. In practice therefore k is initially set to a small value. k is increased only in the neighbourhood of the optimum.

Directly maximising the training set accuracy will by definition, result in the best training set performance possible given the limitations of the model. The model's representational flexibility is directed to this one aim. None of it is modelling other aspects.

This can easily be seen by comparing D and I for a particular training utterance $O(u)$. D involves the probability of observing $O(u)$ from two transcriptions. The transcriptions, other than the correct and best alternative are irrelevant. I however, involves all alternatives. In addition D yields decreasing marginal rewards/penalties for increasingly correct/incorrect classification.

Because of the law of large numbers, for asymptotically large training sets this guarantees a good test set performance. This good performance is achieved without any model correctness assumptions.

It should be noted that direct training set error minimisation is a double edged sword. Very little information from the training set is used. If our HMMs are even approximately correct, and training data limited, this will be wasteful. To understand why this approach may not be ideal with limited training sets, imagine the following situation. Let us assume we have a recogniser which operates by storing the first element of the first observation vector $[O(u)_1]_1$, with its associated transcription $w(u)$. Clearly, unless the training set size is essentially infinite, this system will have perfect training set performance. On the other hand its test set performance will be no better than chance. This behaviour would not be detectable from the only measure the computer had, the value of the CFM objective function. In practice, the question of whether the performance is good given limited training data and a HMM model has to be answered by experiment.

More seriously, differential learning makes strong assumptions about the final use of the models. For CFM if we move to a task with differing prior probabilities or if errors are penalised differently, CFM becomes useless. Methods which learn probability distributions do not suffer from this problem.

3.6 Summary

In this chapter we have studied a number of objective functions. We showed that ML estimators enjoy a number of provably optimal properties. We then presented 3 examples of ML estimators, speech literature ML, MMI and FD. Each of these objective functions enjoy different advantages. Speech literature ML is easily optimised with the Baum-Welch algorithm and it generalises well to an independent test set, even when the training data is limited; FD has a better training set performance than ML and like ML generalises well; and MMI yields optimal performance when the training database is extremely large. Finally, we described one non-ML objective function CFM. When the performance measure of the final system is unweighted string error rate and we have a large training database, it is demonstrably optimal.

A good objective function is only part of the recipe for a good training procedure. Another element which is just as important, is the optimisation algorithm that will be used. This will be studied in the next chapter.

Chapter 4

Optimisation

In chapter 3 we presented various objective functions and reasons for their possible utility. In this chapter we describe algorithms we can use to maximise these functions.

We have already examined the Baum-Welch algorithm for the production of ML models. This algorithm is theoretically guaranteed to converge to a stationary point. Experimentally this convergence is fast. Because of these factors the ML models in this work were trained with this algorithm.

Unfortunately the Baum-Welch algorithm cannot be applied to any of the alternative objective functions we wish to investigate. In fact no algorithm is yet known that gives us both global convergence and speed. Instead heuristic methods must be used. One important source of heuristics is the study of the quadratic function.

It is unlikely that optimisation would be used to solve a quadratic function. This being equivalent to solving a set of linear equations. But its study is important for a number of reasons. The quadratic function is the simplest that can be maximised. Linear functions are either constant or unbounded. In addition Taylor's theorem tells us that many functions can be approximated locally by a quadratic function.

By Taylor's theorem if certain assumptions are satisfied, the objective function F of n-vector λ can be approximated by,

$$F(\lambda) \approx (\lambda - \lambda_0)^T \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_0} + \frac{1}{2}(\lambda - \lambda_0)^T \left.\frac{\partial^2 F}{\partial \lambda^2}\right|_{\lambda_0} (\lambda - \lambda_0) \tag{4.1}$$

Differentiating we obtain,

$$\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_1} \approx \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_0} + \left.\frac{\partial^2 F}{\partial \lambda^2}\right|_{\lambda_0} (\lambda_1 - \lambda_0) \tag{4.2}$$

A necessary condition for a maximum at λ_1 is,

$$\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_1} = 0 \tag{4.3}$$

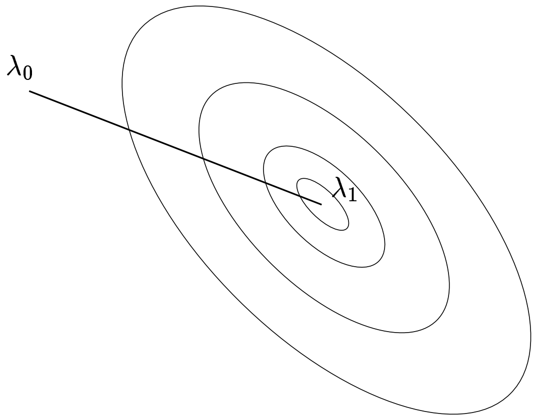

Figure 4.1: Convergence of Newton's algorithm on a quadratic

which yields, if we assume $\left.\frac{\partial^2 F}{\partial \lambda^2}\right|_{\lambda_0}$ is full rank,

$$\lambda_1 = \lambda_0 - \left[\left.\frac{\partial^2 F}{\partial \lambda^2}\right|_{\lambda_0} \right]^{-1} \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_0} \tag{4.4}$$

So starting from an initial approximation λ_0, we can reach the optima λ_1 in one step. This method which is known as Newton's algorithm, is illustrated in figure 4.1. However this algorithm is not recommended because

1. Calculating $\left[\left.\frac{\partial^2 F}{\partial \lambda^2}^{-1}\right|_{\lambda_0} \right]^{-1}$ requires order n^3 time and order n^2 space.
2. The update suggested by 4.4 may be so large that it makes the approximation 4.1 invalid.
3. The presence of minima and points of inflection if sufficiently near to λ_0 will make the iteration unstable. It may in these circumstances step to infinity, converge to these unwanted points or increase the number of iterations required.
4. $\left.\frac{\partial^2 F}{\partial \lambda^2}\right|_{\lambda_0}$ may not be full rank and so may not be invertable.

Even though this update is not directly usable, the quadratic model and the form of its solution is the heuristic basis of many algorithms.

One important concept when minimising the quadratic approximation is that of conjugacy. If $\frac{\partial^2 F}{\partial \lambda^2}$ is full rank, it can be shown that there exists a n by n full rank matrix P such that,

$$P^T \left.\frac{\partial^2 F}{\partial \lambda^2}\right|_{\lambda_0} P = D \tag{4.5}$$

where D is a diagonal matrix. There are an infinite number of choices for P. For instance, the columns of P could be eigenvectors of $\frac{\partial^2 F}{\partial \lambda^2}$. If we make the transformation

$$\lambda - \lambda_0 = Py \tag{4.6}$$

then,

$$F \approx y^T P^T \left. \frac{\partial F}{\partial \lambda} \right|_{\lambda_0} + \frac{1}{2} y^T D y \tag{4.7}$$

The maximum of 4.7 can be found by maximising with respect to each element in y separately. This is identical to sequentially maximising F along the vectors $P_1, P_2, \ldots, P_n$, where P_j is the column j from P. These directions are known as a conjugated set of directions.

This procedure has the property of quadratic termination. That is the method is guaranteed to find the optima for a quadratic in n iterations.

Considerations like size of the optimisation problem and the relative difficulty of calculating function, gradient and hessian information serve to restrict our choice of optimisation algorithm. Our quadratic approximation contains $n(n+3)/2$ independent terms. Altering any one of these will alter the position of the maximum. We should not expect to find the exact maximum until we have calculated this much information.

For our recognition tasks we have an n in the tens of thousands. We do not have any chance of finding the exact maximum. The best we can hope for is to come close before our resources run out. We therefore use the heuristic that we should maximise the rate at which we calculate information. Our optimisation experiments will search for an algorithm that is effective in utilising this information.

We can immediately rule out methods that compute the hessian directly. This requires of order n^2 space, so it is impossible to store. As we shall see later, gradient information is almost free once the function value has been calculated. It therefore makes sense to restrict our optimisation study to gradient methods.

4.1 Constraints

During maximisation we must always maintain constraints 2.1, 2.3 and 2.5. In our experiments with the MMI objective function however, constraints 2.2 and 2.4 were ignored. Although this removes the original probabilistic interpretations of the resulting transitions and mixture weights, our Viterbi decoding algorithm remains valid. This relaxation also leaves the probabilistic constraints on the classification statistic $P(w|O)$ intact. Initial experiments showed that maintaining constraint 2.2 and 2.4 did not produce a consistent change of performance. With the FD objective function constraint 2.2 needs to be imposed. Without it the FD objective function becomes unbounded. As in the MMI case, in our main experiments we ignored constraint 2.4. The probabilistic nature of the $P(s|O(u))$ estimate does not require it. Again there was no consistent change of performance when constraint 2.4 was imposed.

There are a number of methods that impose constraints. Most can be ruled out because at any stage the constraints are only approximately satisfied. In addition, many involve complicated extra stages in an already complexity limited process. One method that does not have these failings is transformation. Because constraint satisfaction was not the main thrust of this work, we limited ourselves to this technique.

In the transform method instead of maximising with respect to variables $\{a_{i,j}, m_{i,k}, C_{i,k}, w_{i,k}\}$, we maximise with respect to $\{a'_{i,j}, m_{i,k}, C'_{i,k}, w'_{i,k}\}$. These two domains are related by

$$a_{i,j} = \begin{cases} \frac{a'^2_{i,j}}{\sum_{k=1}^{N} a'^2_{i,k}} & \text{if constraint 2.2 is imposed,} \\ a'^2_{i,j} & \text{otherwise.} \end{cases} \tag{4.8}$$

$$w_{i,k} = \begin{cases} \frac{w'^2_{i,k}}{\sum_{j=1}^{N} w'^2_{i,j}} & \text{if constraint 2.4 is imposed,} \\ w'^2_{i,k} & \text{otherwise.} \end{cases} \tag{4.9}$$

$$C_{i,k} = C'_{i,k} C'^T_{i,k} \tag{4.10}$$

It can easily be verified that for every $\{a'_{i,j}, w'_{i,k}\}$, the resulting $\{a_{i,j}, w_{i,k}\}$ do mathematically satisfy our constraints. However, $C'_{i,k}$ only ensures non-negative definiteness. Our tools incorporate a command-line option which controls the addition of a positive definite diagonal matrix to our $C_{i,k}$ reestimate. Similar command-line options also exist to floor $a_{i,j}$ and $w_{i,k}$ reestimates, however, these command-line options were not required in any of our experiments.

4.2 Derivatives

In this section we present expressions for the derivatives of our objective functions.

We will first consider the MMI objective function.

$$\frac{\partial I}{\partial \lambda} = \frac{\partial}{\partial \lambda} \sum_{u=1}^{U} \log P_\lambda(w(u)|O(u)) \tag{4.11}$$

$$= \sum_{u=1}^{U} \left[\frac{\partial \log P_\lambda(O(u)|w(u)) P^*(w(u))}{\partial \lambda} - \frac{\partial \log P_\lambda(O(u)|R)}{\partial \lambda} \right] \tag{4.12}$$

$$= \sum_{u=1}^{U} \frac{\partial \log P_\lambda(O(u)|w(u))}{\partial \lambda} - \sum_{u=1}^{U} \frac{\partial \log P_\lambda(O(u)|R)}{\partial \lambda} \tag{4.13}$$

The derivative consists of two terms corresponding to the transcription and recognition models. Below we shall drop the dependency on $w(u)$ or R. The reader should remember that the correct HMM must be used when calculating the required quantities.

In addition to 2.38 and 2.42 we require the derivatives of the constraint equations 2.1, 2.2, 2.3, 2.4 and 2.5.

If a sum to one constraint is imposed,

$$\frac{\partial a_{i,j}}{\partial a'_{i,p}} = \frac{2a'_{i,j}\delta_{p,j} - 2a'_{i,p}a_{i,j}}{\sum_{k=1}^{N} a'^2_{i,k}} \tag{4.14}$$

$$\frac{\partial w_{i,k}}{\partial w'_{i,p}} = \frac{2w'_{i,k}\delta_{p,k} - 2w'_{i,p}w_{i,k}}{\sum_{k=1}^{N} w'^2_{i,k}} \tag{4.15}$$

If on the other hand, there is no sum to one constraint,

$$\frac{\partial a_{i,j}}{\partial a'_{i,p}} = 2a'_{i,j}\delta_{p,j} \tag{4.16}$$

$$\frac{\partial w_{i,k}}{\partial w'_{i,p}} = 2w'_{i,k}\delta_{p,k} \tag{4.17}$$

In addition we need the observation derivatives [22],

$$\frac{\partial \log b_{i,k}(O(u)_t)}{\partial m_{i,k}} = C_{i,k}^{-1}(O(u)_t - m_{i,k}) \tag{4.18}$$

$$\frac{\partial \log b_{i,k}(O(u)_t)}{\partial C'_{i,k}} = -C'^{-T}_{i,k} + C_{i,k}^{-1}(O(u)_t - m_{i,k})(O(u)_t - m_{i,k})^T C'^{-T}_{i,k} \tag{4.19}$$

We now have the necessary components for our results.
If the sum to one constraints are imposed,

$$\frac{\partial \log P_\lambda(O(u))}{\partial a'_{i,p}} = \sum_{j=1}^{N} \frac{\partial \log P_\lambda(O(u))}{\partial a_{i,j}} \frac{\partial a_{i,j}}{\partial a'_{i,p}} \tag{4.20}$$

$$= \left(\frac{2a'_{i,N}\delta_{p,N} - 2a'_{i,p}a_{i,N}}{a'^2_{i,N}}\right) P_\lambda(s_{T(u)} = i|O(u)) + \sum_{j=1}^{N-1}\left(\frac{2a'_{i,j}\delta_{p,j} - 2a'_{i,p}a_{i,j}}{a'^2_{i,j}}\right)\sum_{t=0}^{T(u)-1} P_\lambda(s_t = i, s_{t+1} = j|O(u)) \tag{4.21}$$

$$\frac{\partial \log P_\lambda(O(u))}{\partial w'_{i,p}} = \sum_{k=1}^{K} \frac{\partial \log P_\lambda(O(u))}{\partial w_{i,k}} \frac{\partial w_{i,k}}{\partial w'_{i,p}} \tag{4.22}$$

$$= \sum_{k=1}^{K}\left(\frac{2w'_{i,k}\delta_{p,k} - 2w'_{i,p}w_{i,k}}{w'^2_{i,k}}\right)\sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u)) \tag{4.23}$$

If there is no sum to one constraint,

$$\frac{\partial \log P_\lambda(O(u))}{\partial a'_{i,p}} = \sum_{j=1}^{N} \frac{\partial \log P_\lambda(O(u))}{\partial a_{i,j}} \frac{\partial a_{i,j}}{\partial a'_{i,p}} \tag{4.24}$$

$$= \begin{cases} \frac{2}{a'_{i,p}} \sum_{t=0}^{T(u)-1} P_\lambda(s_t = i, s_{t+1} = p|O(u)) & p \neq N \\ \frac{2}{a'_{i,N}} P_\lambda(s_{T(u)} = N, |O(u)) & p = N \end{cases} \tag{4.25}$$

$$\frac{\partial \log P_\lambda(O(u))}{\partial w'_{i,p}} = \frac{2}{w'_{i,p}} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = p|O(u)) \tag{4.26}$$

Using 4.18 and 4.19 the derivatives with respect to the observation parameters are,

$$\frac{\partial \log P_\lambda(O(u))}{\partial m_{i,k}} = C_{i,k}^{-1} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k | O(u))(O(u)_t - m_{i,k}) \tag{4.27}$$

$$\frac{\partial \log P_\lambda(O(u))}{\partial C'_{i,k}} = \sum_{t=1}^{T(u)} \frac{P_\lambda(O(u))}{\partial \log b_{i,k}(O(u)_t)} \frac{\partial \log b_{i,k}(O(u)_t)}{\partial C'_{i,k}} \tag{4.28}$$

$$= -\frac{1}{2} C'^{-T}_{i,k} \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k | O(u)) + C_{i,k}^{-1} \left[\sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k | O(u))(O(u)_t - m_{i,k})(O(u)_t - m_{i,k})^T \right] C'^{-T}_{i,k} \tag{4.29}$$

The derivative equations 4.21,4.23,4.25,4.26,4.27 and 4.29 all involve two types of components. The first type contains the current HMM parameters $\{a_{i,j}, m_{i,k}, c_{i,k}, w_{i,k}\}$. Obviously these are immediately available without any computation. The second type contains an assortment of expected values. These are immediately available if $P_\lambda(O(u))$ was calculated using the Baum-Welch algorithm. HMMs are therefore unusual in that once the probability has been found, the derivatives can be calculated with very little extra work.

4.3 Steepest Ascent

This algorithm [4, 5], uses the following update

$$\lambda_{k+1} = \lambda_k + \alpha_k \left. \frac{\partial F}{\partial \lambda} \right|_{\lambda_k} \tag{4.30}$$

In effect we have replaced the hessian term in the heuristic equation 4.4 with the identity matrix. The learning rate α_k, can either be fixed by the user or more expensively found by a line search (see 4.10).

For small α_k the method traces out an ascending path in parameter space, along the line of greatest slope. In practice this algorithm often displays a number of faults. By ignoring the second and higher order derivatives the region of validity of the update equation is small. The line of greatest slope, although locally appealing, may lead to only small improvements. Successive gradients if exact line searches are used, are orthogonal not conjugate. In general later steps may introduce gradient components along already searched directions. This is illustrated in figure 4.2. This method is not quadratically convergent, but if good line searches are used, the method is globally convergent.

4.4 On-line Methods

The objective functions that we wish to study consist of a sum of similar terms. It is often the case that there is some redundancy in the training set. That is, the additional marginal

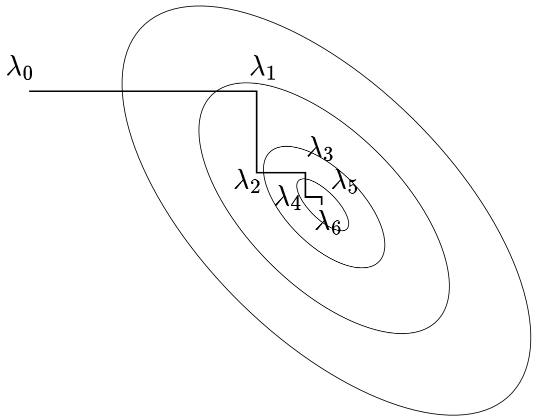

Figure 4.2: Convergence of steepest ascent on a quadratic

information reduces as each term is added. Symbolically the situation is,

$$F = \sum_{u=1}^{U} \log F_u \tag{4.31}$$

$$\approx \log F_u \tag{4.32}$$

If we duplicate the training data 10 times, the work required by equation 4.30 increases by the same factor. Clearly we would do better by updating 10 times more often. This process taken to the extreme, when we update every training pattern, leads to an algorithm known as stochastic approximation, [2, 5, 77].

The update equation now becomes

$$\lambda_{n+1} = \lambda_n + \alpha_n \left. \frac{\partial \log \hat{F}}{\partial \lambda} \right|_{\lambda_n} \tag{4.33}$$

where $\frac{\partial \log \hat{F}}{\partial \lambda}$ is the gradient on the n^{th} randomly chosen training example.

It can be shown that the algorithm converges to a stationary point with probability one, if α_n satisfies,

$$\lim_{n \to \infty} \alpha_n = 0 \tag{4.34}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty \tag{4.35}$$

$$\sum_{n=0}^{\infty} \alpha_n^2 < \infty \tag{4.36}$$

Unfortunately in the real world we must terminate training after a finite amount of time. We therefore may find that an update strategy between 4.30 and 4.33 may be best.

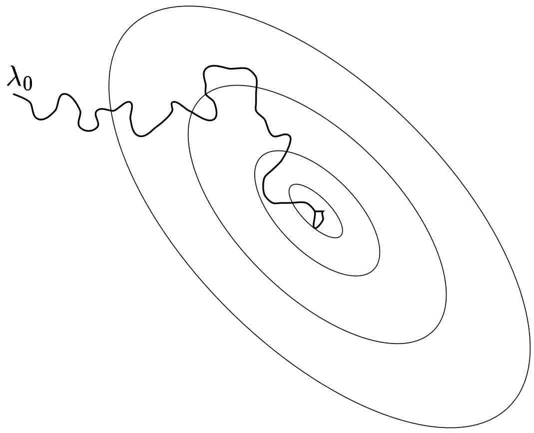

Figure 4.3: Convergence of stochastic approximation on a regular maximum

The effectiveness of on-line methods depends on how close to the maximum we wish to come; the data redundancy; the objective function and the model. How it compares with other optimisation algorithms must be examined experimentally.

The behaviour of stochastic approximation on a quadratic function is illustrated in figure 4.3.

4.5 Conjugate Gradients

A major failing of steepest ascent is that parameter updates may introduce gradient components along already searched directions. The notion of conjugate directions avoids this difficulty (see page 36). The conjugate gradient method seeks to find such directions efficiently [4, 5, 46, 80]. Instead of using pure gradient as the search direction, conjugate gradients removes those components which are not conjugate to previous search directions. That is, the search directions are given by:

$$d_i = \begin{cases} \frac{\partial F}{\partial \lambda}\Big|_{\lambda_0} & i = 0, \\ \frac{\partial F}{\partial \lambda}\Big|_{\lambda_i} + \sum_{j=0}^{i-1} Z_{i,j} d_j & \text{otherwise.} \end{cases} \tag{4.37}$$

where d_i is the search direction from point λ_i.

The coefficients $Z_{i,j}$ are chosen assuming constant $\frac{\partial^2 F}{\partial \lambda^2}$, so that d_i satisfies the conjugacy conditions.

$$d_i^T \frac{\partial^2 F}{\partial \lambda^2} d_j = 0 \quad \text{for } i \neq j \tag{4.38}$$

By applying $d_j^T \frac{\partial^2 F}{\partial \lambda^2}$ to 4.37 we obtain,

$$Z_{i,j} = -\frac{d_j^T \frac{\partial^2 F}{\partial \lambda^2} \frac{\partial F}{\partial \lambda}\Big|_{\lambda_i}}{d_j^T \frac{\partial^2 F}{\partial \lambda^2} d_j} \tag{4.39}$$

But we know from 4.2

$$\alpha \frac{\partial^2 F}{\partial \lambda^2} d_j = \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{j+1}} - \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_j} \tag{4.40}$$

Substituting into 4.39 we have,

$$Z_{i,j} = -\frac{\left(\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{j+1}} - \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_j}\right)^T \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i}}{\left(\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{j+1}} - \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_j}\right)^T d_j} \tag{4.41}$$

But 4.37 gives us

$$\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_k} = d_k - \sum_{j=0}^{k-1} Z_{k,j} d_j \tag{4.42}$$

If the maximisation's along the previous search directions have been exact then,

$$\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i} d_k = 0 \quad k = 1, \ldots, i-1 \tag{4.43}$$

4.42 and 4.43 allows us to prove, if $k < i$, the orthogonality of $\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i}$ and $\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_k}$.

$$\begin{aligned} \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i}^T \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_k} &= \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i}^T \left(d_k - \sum_{j=0}^{k-1} Z_{k,j} d_j\right) && (4.44)\\ &= 0 && (4.45) \end{aligned}$$

This result allows us to present our final expressions for $Z_{i,j}$. All are equivalent for a quadratic function and perfect line searches. In practice, however, their values differ. Using 4.45 and 4.41 we obtain,

$$Z_{i,j} = \begin{cases} 0 & \text{for } j < i-1, \\ -\frac{\left(\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i} - \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{i-1}}\right)^T \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i}}{\left(\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i} - \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{i-1}}\right)^T d_{i-1}} & \text{otherwise.} \end{cases} \tag{4.46}$$

which is known as the Beale-Sorenson expression.

Alternatively, substituting 4.37 into 4.46 we obtain,

$$Z_{i,j} = \begin{cases} 0 & \text{for } j < i-1, \\ -\frac{\left(\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i} - \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{i-1}}\right)^T \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_i}}{\left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{i-1}} \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda_{i-1}}} & \text{otherwise.} \end{cases} \tag{4.47}$$

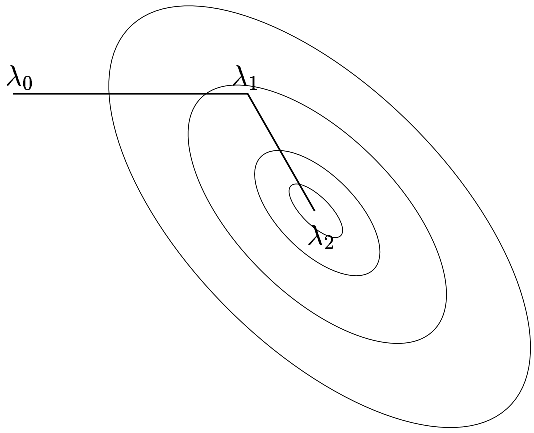

Figure 4.4: Convergence of conjugate gradients on a quadratic

the Polak-Ribiere expression. Using 4.45 in 4.47 we obtain the Fletcher-Reeves expression,

$$Z_{i,j} = \begin{cases} 0 & \text{for } j < i-1, \\ -\frac{\frac{\partial F}{\partial \lambda}\big|_{\lambda_i} \frac{\partial F}{\partial \lambda}\big|_{\lambda_i}}{\frac{\partial F}{\partial \lambda}\big|_{\lambda_{i-1}} \frac{\partial F}{\partial \lambda}\big|_{\lambda_{i-1}}} & \text{otherwise.} \end{cases} \tag{4.48}$$

Summarising the conjugate gradient algorithm is given by

$$d_i = \begin{cases} \frac{\partial F}{\partial \lambda}\big|_{\lambda_0} & i = 0 \\ \frac{\partial F}{\partial \lambda}\big|_{\lambda_i} + Z_{i,i-1} d_{i-1} & \text{otherwise.} \end{cases} \tag{4.49}$$

$$\lambda_{i+1} = \lambda_i + \alpha_i d_i \tag{4.50}$$

where $Z_{i,j}$ is given by 4.46, 4.47 or 4.48 and α_i, is chosen to maximise F along direction d_i.

For a quadratic function, exact arithmetic and line searches this algorithm exhibits quadratic termination. But the error surfaces in our tasks do not have these properties. However, there is still the hope that this method will do better than steepest ascent.

The behaviour of this algorithm on a quadratic function is illustrated in figure 4.4

4.6 Quick-Prop

In this algorithm we replace the hessian term in 4.4 with a finite difference diagonal approximation [23, 46]. That is,

$$\left[\frac{\partial^2 F}{\partial \lambda^2}\right]_{i,j} = \begin{cases} 0 & \text{for } i \neq j \\ \frac{\left[\frac{\partial F}{\partial \lambda}\big|_{\lambda_n}\right]_i - \left[\frac{\partial F}{\partial \lambda}\big|_{\lambda_{n-1}}\right]_i}{[\lambda_n]_i - [\lambda_{n-1}]_i} & \text{otherwise} \end{cases} \tag{4.51}$$

where $[m]_{i,j}$ is element (i,j) of matrix m

Substituting this approximation into 4.4 we have

$$[\lambda_{n+1}]_i = [\lambda_n]_i - \frac{([\lambda_n]_i - [\lambda_{n-1}]_i)}{\left(\left[\frac{\partial F}{\partial \lambda}\Big|_{\lambda_n}\right]_i - \left[\frac{\partial F}{\partial \lambda}\Big|_{\lambda_{n-1}}\right]_i\right)} \left[\frac{\partial F}{\partial \lambda}\Big|_{\lambda_n}\right]_i \tag{4.52}$$

where $[v]_i$ is element i of vector v.

As with all, heuristic iterative optimisation algorithms, we must impose some restrictions on this update rule to ensure stability. As suggested in [23], we ensure the following, that the step proceeds uphill,

$$([\lambda_{n+1}]_i - [\lambda_n]_i) \left[\frac{\partial F}{\partial \lambda}\Big|_{\lambda_n}\right]_i > 0 \tag{4.53}$$

and that the step is small enough to be trusted.

$$([\lambda_{n+1}]_i - [\lambda_n]_i) < G\left([\lambda_n]_i - [\lambda_{n-1}]_i\right) \tag{4.54}$$

where G, the growth factor, is a parameter set by the user. Its value must be greater than 1.0.

Equation 4.53 can only be violated if our finite difference hessian component 4.51 is positive. In this situation our update equation attempts to move to the critical point at the center of the nearby minimum. Instead our quadratic model suggests the correct course of action is to move a large distance up the gradient. If equation 4.54 is violated, prudency suggests we should limit the growth of the step to G. In both circumstances, we can use the following default update.

$$[\lambda_{n+1}]_i = [\lambda_n]_i + G\,\mathrm{Abs}\left([\lambda_n]_i - [\lambda_{n-1}]_i\right) \mathrm{Sign}\left(\left[\frac{\partial F}{\partial \lambda}\Big|_{\lambda_n}\right]_i\right) \tag{4.55}$$

To bootstrap this algorithm, we use for the first iteration either steepest ascent, or the algorithm introduced in section 4.8, Manhattan learning.

We have with this algorithm abandoned hope of finding the global maximum (except in the exceptional cases). Instead we attempt to make as much progress as possible before resources are exhausted.

4.7 On-line Quick-Prop

With a view to speeding up convergence on highly redundant training sets, quick-prop was transformed into an on-line algorithm. After some brief experimentation the following scheme was chosen.

The training data was split into P approximately equal sized blocks. The differing block sizes were compensated for by dividing the function values and gradients by the number of frames in each block. An update occurred after each block was processed. It was found that better convergence was obtained if the hessian approximation involved only one block of data. Hence, it was calculated as,

$$\left[\frac{\partial^2 F}{\partial \lambda^2}\right]_{i,j} = \begin{cases} 0 & \text{for } i \neq j \\ \frac{\left[\frac{\partial F}{\partial \lambda}\big|_{\lambda_n}\right]_i - \left[\frac{\partial F}{\partial \lambda}\big|_{\lambda_{n-P}}\right]_i}{[\lambda_n]_i - [\lambda_{n-P}]_i} & \text{otherwise} \end{cases} \tag{4.56}$$

To ensure stability it was also found to be necessary to reduce the maximum growth-factor G in equation 4.54. Because x_{n-P} and $\frac{\partial F}{\partial \lambda}\big|_{\lambda_{n-P}}$ are not available during the first P updates the normal quick-prop algorithm is used for these updates.

The rest of the original quick-prop algorithm remains unchanged.

4.8 Manhattan Optimisation

In this section we present a learning algorithm which uses a Manhattan update rule. In contrast to the methods described previously, the update does not involve the gradient magnitude. Instead it uses a fixed update, the size being chosen to hopefully move us to a maximum.

To obtain the update step we have to make a number of approximations. These make our expressions easily realisable and hopefully still useful.

Our approximations are:

1. When calculating the update step we replace our objective function with log probability. We hope that the shape of the two functions are similar.

2. We use a 2nd order Taylor series to calculate our update. That is, we use Newton's method.

3. We assume constant segmentation when calculating the hessian.

4. We assume the hessian is diagonal.

5. At various points we either replace sample statistics with related quantities or ignore them.

The Newton update involves the second order derivatives. By analogy to 2.38, our approximation for these quantities are,

$$\frac{\partial^2 F}{\partial \lambda^2} \approx \frac{\partial^2 Q(\lambda, \overline{\lambda})}{\partial \overline{\lambda}^2}\bigg|_{\overline{\lambda}=\lambda} \tag{4.57}$$

$$= \sum_{u=1}^{U} \sum_{m \in w(u)} P_\lambda(m|O(u), w(u)) \frac{\partial^2 \log P_\lambda(m, O(u)|w(u))}{\partial \lambda^2} \tag{4.58}$$

For our presentation we shall split up the parameter vector λ, into the various component types $a'_{i,j}, m_{i,k}, C'_{i,k}, w'_{i,k}$. The reader is reminded that $a'_{i,j}, C'_{i,k}, w'_{i,k}$ were introduced in order to remove our parameter constraints. In order to simplify implementation we ignored the transition and weight sum to one constraints and we assume diagonal covariance gaussians. These simplifications were deemed to be acceptable because we are attempting to approximate second order training information and because of the gross approximations already made.

We shall now derive the Manhattan update of $a_{i,j}, j \neq N$. Differentiating 4.25 we obtain,

$$\frac{\partial^2 F}{\partial a'^2_{i,j}} \approx -\frac{2}{a'^2_{i,j}} \sum_{u=1}^{U} \sum_{t=0}^{T(u)-1} P_\lambda(s_t = i, s_{t+1} = j|O(u), w(u)) \tag{4.59}$$

Therefore,

$$-\left(\frac{\partial^2 F}{\partial a'^2_{i,j}}\right)^{-1}\left(\frac{\partial F}{\partial a'_{i,j}}\right) \approx a'_{i,j} \quad j \neq N \tag{4.60}$$

Analogously we can obtain,

$$-\left(\frac{\partial^2 F}{\partial a'^2_{i,N}}\right)^{-1}\left(\frac{\partial F}{\partial a'_{i,N}}\right) \approx a'_{i,N} \tag{4.61}$$

$$-\left(\frac{\partial^2 F}{\partial w'^2_{i,k}}\right)^{-1}\left(\frac{\partial F}{\partial w'_{i,k}}\right) \approx w'_{i,k} \tag{4.62}$$

Differentiating 4.27 we obtain,

$$\frac{\partial^2 F}{\partial m^2_{i,k}} \approx -C^{-1}_{i,k}\sum_{u=1}^{U}\sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u)) \tag{4.63}$$

Therefore the Manhattan mean update is,

$$-\left(\frac{\partial^2 F}{\partial m^2_{i,k}}\right)^{-1}\left(\frac{\partial F}{\partial m_{i,k}}\right) \approx \frac{\sum_{u=1}^{U}\sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))(O(u)_t - m_{i,k})}{\sum_{u=1}^{U}\sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))} \tag{4.64}$$

If the ML means and variances are approximately correct for the optima of the new objective function F then,

$$[O(u)_t - m_{i,k}]_j \approx \left[C'_{i,k}\right]_{j,j} \tag{4.65}$$

We can simplify the Manhattan update to,

$$\left[-\left(\frac{\partial^2 F}{\partial m^2_{i,k}}\right)^{-1}\left(\frac{\partial F}{\partial m_{i,k}}\right)\right]_j \approx \left[C'_{i,k}\right]_{j,j} \tag{4.66}$$

Finally we present the Manhattan update for $\left[C'_{i,k}\right]_{j,j} = \sigma_j$. Differentiating 4.29 we obtain,

$$\frac{\partial^2 F}{\partial \sigma^2_j} \approx \sum_{u=1}^{U}\sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k|O(u), w(u))\left[\frac{1}{\sigma^2_j} - \frac{3\left([O(u)_t - m_{i,k}]_j\right)^2}{\sigma^4_j}\right] \tag{4.67}$$

If we ignore the $\left([O(u)_t - m_{i,k}]_j\right)^2$ term in 4.67 and in 4.29 we make the assumption that,

$$\left([O(u)_t - m_{i,k}]_j\right)^2 \approx \sigma^2_j \tag{4.68}$$

we obtain,

$$-\left(\frac{\partial^2 F}{\partial \sigma_j^2}\right)^{-1}\left(\frac{\partial F}{\partial \sigma_j}\right) \approx \left[C'_{i,k}\right]_{j,j} \tag{4.69}$$

In practice the above steps are too large. Hence our actual updates will be a small fraction of the above expressions.

Summarising, our updates are:

$$\overline{a'}_{i,j} = a'_{i,j} + \epsilon^a a'_{i,j} \operatorname{Sign}\left(\frac{\partial F}{\partial a'_{i,j}}\right) \tag{4.70}$$

$$\overline{w'}_{i,k} = w'_{i,k} + \epsilon^w w'_{i,k} \operatorname{Sign}\left(\frac{\partial F}{\partial w'_{i,k}}\right) \tag{4.71}$$

$$\left[\overline{m}_{i,k}\right]_j = \left[m_{i,k}\right]_j + \epsilon^m \left[C'_{i,k}\right]_{j,j} \operatorname{Sign}\left(\frac{\partial F}{\partial \left[m_{i,k}\right]_j}\right) \tag{4.72}$$

$$\left[\overline{C'}_{i,k}\right]_{j,j} = \left[C'_{i,k}\right]_{j,j} + \epsilon^C \left[C'_{i,k}\right]_{j,j} \operatorname{Sign}\left(\frac{\partial F}{\partial \left[C'_{i,k}\right]_{j,j}}\right) \tag{4.73}$$

where $\epsilon^a, \epsilon^w, \epsilon^m, \epsilon^C$ are the parameter dependent learning rates.

For the full-covariance case we also require a suitable update for the off-diagonal covariance terms. Unfortunately, there does not appear to be a simple derivable expression for it. Instead the following was selected by analogy to 4.70, 4.71, 4.72 and 4.73.

$$\left[\overline{C'}_{i,k}\right]_{l,m} = \left[C'_{i,k}\right]_{l,m} + \epsilon^C \frac{\sqrt{\left[C'_{i,k}\right]_{l,l}\left[C'_{i,k}\right]_{m,m}}}{2} \operatorname{Sign}\left(\frac{\partial F}{\partial \left[C'_{i,k}\right]_{l,m}}\right) \tag{4.74}$$

The simplicity of these expressions is striking. The updates are also independent of differing variance and occupancy scale effects. Apart from the sign of the gradients there are no dynamically calculated statistics involved. This makes the method a good candidate for use in an on-line strategy.

4.9 Extended Baum-Welch

In recent studies of MMI estimation the extended Baum-Welch algorithm has become very popular. However, it has none of the theoretical guarantees of ML Baum-Welch. That is, it does not guarantee convergence to a stationary point. In practice though, it has been shown to be useful [14, 30, 31, 60–63].

The derivation of extended Baum-Welch from ML Baum-Welch is long and not very informative. Because of this the reader is referred to [60] for details. We shall limit ourselves to the presentation of the final re-estimation equations. (The reader should note that the variance update has been extended to the full covariance case. By experiment [47], this extension has been shown to be useful).

This algorithm can be applied to the FD objective function without change. The update equations are:

$$\overline{a}_{i,j} = \frac{a_{i,j}\left[D^a_{i,j} + E^a\right]}{\sum_{k=1}^N a_{i,k}\left[D^a_{i,k} + E^a\right]} \tag{4.75}$$

$$\overline{w}_{i,k} = \frac{w_{i,k}\left[D^w_{i,k} + E^w\right]}{\sum_{j=1}^K w_{i,j}\left[D^w_{i,j} + E^w\right]} \tag{4.76}$$

$$\overline{m}_{i,k} = \frac{\sum_{u=1}^U \sum_{t=1}^{T(u)} \psi^{u,t}_{i,k} O(u)_t + F m_{i,k}}{\sum_{u=1}^U \sum_{t=1}^{T(u)} \psi^{u,t}_{i,k} + F} \tag{4.77}$$

$$\overline{C}_{i,k} = \frac{\sum_{u=1}^U \sum_{t=1}^{T(u)} \psi^{u,t}_{i,k} O(u)_t O(u)_t^T + F\left(C_{i,k} + m_{i,k} m_{i,k}^T\right)}{\sum_{u=1}^U \sum_{t=1}^{T(u)} \psi^{u,t}_{i,k} + F} - \overline{m}_{i,k}\overline{m}_{i,k}^T \tag{4.78}$$

where,

$$C^{a,w}_{i,j} = \begin{cases} \sum_{u=1}^U \sum_{t=1}^{T(u)} P_\lambda(s_t = i, s_{t+1} = j | O(u), w(u)) & j \neq N \\ \sum_{u=1}^U P_\lambda(s_{T(u)} = i | O(u), w(u)) & \text{otherwise.} \end{cases} \tag{4.79}$$

$$C^{a,R}_{i,j} = \begin{cases} \sum_{u=1}^U \sum_{t=1}^{T(u)} P_\lambda(s_t = i, s_{t+1} = j | O(u), R) & j \neq N \\ \sum_{u=1}^U P_\lambda(s_{T(u)} = i | O(u), R) & \text{otherwise.} \end{cases} \tag{4.80}$$

$$D^a_{i,j} = \frac{C^{a,w}_{i,j}}{\sum_{j=1}^N C^{a,w}_{i,j}} - \frac{C^{a,R}_{i,j}}{\sum_{j=1}^N C^{a,R}_{i,j}} \tag{4.81}$$

$$E^a = 1.01 \max_{i,j} \left\{-D^a_{i,j}, 0\right\} \tag{4.82}$$

$$C^{w,w}_{i,k} = \sum_{u=1}^U \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = k | O(u), w(u)) \tag{4.83}$$

$$C^{w,R}_{i,k} = \sum_{u=1}^U \sum_{t=1}^{T(u)} P_\lambda(s_t = i, g_t = j | O(u), R) \tag{4.84}$$

$$D^w_{i,k} = \frac{C^{w,w}_{i,k}}{\sum_{k=1}^K C^{w,w}_{i,j}} - \frac{C^{w,R}_{i,k}}{\sum_{k=1}^K C^{w,R}_{i,k}} \tag{4.85}$$

$$E^w = 1.01 \max_{i,k} \left\{-D^w_{i,k}, 0\right\} \tag{4.86}$$

$$\psi^{u,t}_{i,k} = P_\lambda(S_t = i, g_t = k | O(u), w(u)) - P_\lambda(S_t = i, g_t = k | O(u), R) \tag{4.87}$$

The parameter F was chosen using the method suggested in [60]. Briefly it was made large enough so that equation 4.78 produced positive definite re-estimates. First F was set to a small value. It was then doubled until all of the resulting $\overline{C}_{i,k}$ were positive definite. This final value was finally doubled again to provide a safety margin.

4.10 Line Search

In this section we consider the problem of maximising a generic objective function along a line. That is, we maximise F along the line $\lambda_i + \alpha_i d_i$, where λ_i is the current point and d_i is the given search direction. The resulting algorithm can be used as a component of the steepest ascent 4.3 or conjugate gradient algorithms 4.5.

Our chosen algorithm makes no assumptions about the form of F. In particular it was chosen because it is guaranteed to converge on a local maximum no matter how uncooperative F is. This assurance is achieved by the use of embedded intervals each guaranteed to contain a maxima. Continuity considerations tells us that an interval will contain a maxima if

$$\left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} > 0 \tag{4.88}$$

and,

$$\left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = b} < 0 \tag{4.89}$$

or,

$$F(\lambda_i + a d_i) > F(\lambda_i + b d_i) \tag{4.90}$$

These situations are illustrated in figure 4.5.

Our algorithm has two stages, interval location and interval refinement. A major part of these stages is a bounded cubic interpolation/extrapolation step. The next three subsections describe our cubic fitting, interval location and interval refinement steps in more detail.

4.10.1 Cubic Fitting

This process approximates $F(\lambda_i + \alpha_i d_i)$ as a function of α_i by a cubic. The coefficients of this cubic are chosen so that its value and derivative match those of $F(\lambda_i + \alpha_i d_i)$ at the endpoints of the current interval a and b. In symbolic terms we wish to make the approximation

$$F(\lambda_i + \alpha_i d_i) \approx c_0 + c_1(\alpha_i - a) + c_2(\alpha_i - a)^2 + c_3(\alpha_i - a)^3 \tag{4.91}$$

$$\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i} \approx c_1 + 2c_2(\alpha_i - a) + 3c_3(\alpha_i - a)^2 \tag{4.92}$$

The reader should note that we have chosen the form of our cubic with the aim of simplifying our proofs. Setting α_i to a in 4.91 and 4.92 we obtain,

$$c_0 = F(\lambda_i + a d_i) \tag{4.93}$$

$$c_1 = \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} \tag{4.94}$$

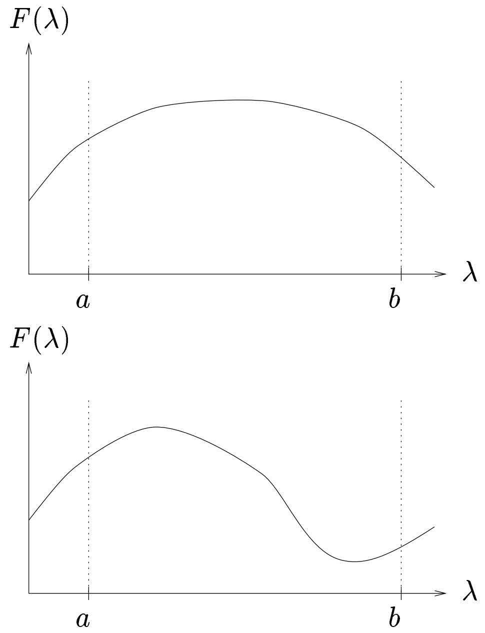

Figure 4.5: Intervals guaranteed to bracket maxima

Substituting 4.93 and 4.94 into 4.91 and 4.92, and setting α_i to b we obtain,

$$c_2(b-a)^2 + c_3(b-a)^3 = F(\lambda_i + bd_i) - F(\lambda_i + ad_i) - \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} (b-a) \tag{4.95}$$

$$2c_2(b-a) + 3c_3(b-a)^2 = \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = b} - \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} \tag{4.96}$$

Solving these for $c_2(b-a)$ and $c_3(b-a)^2$ we obtain,

$$c_2(b-a) = -\left(\left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} + x \right) \tag{4.97}$$

$$c_3(b-a)^2 = \frac{1}{3}\left(\left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} + \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = b} + 2x \right) \tag{4.98}$$

where,

$$x = \frac{3}{(b-a)}\left(F(\lambda_i + ad_i) - F(\lambda_i + bd_i)\right) + \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = a} + \left.\frac{\partial F(\lambda_i + \alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i = b} \tag{4.99}$$

4.92 now becomes

$$\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i} \approx \left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} - 2\left(\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + x\right)\frac{(\alpha_i-a)}{(b-a)} + \left(\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + \left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=b} + 2x\right)\frac{(\alpha_i-a)^2}{(b-a)^2} \tag{4.100}$$

The critical points of our cubic $\alpha^{\pm}$, can be found by setting 4.100 to zero and solving for α_i.

$$\alpha^{\pm} = a + (b-a)\left(\frac{\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + x \pm y}{\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + \left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=b} + 2x}\right) \tag{4.101}$$

where,

$$y = \sqrt{x^2 - \left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a}\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=b}} \tag{4.102}$$

The answer to which of these roots $\alpha^{\pm}$ is the maximum, can be found by examining the sign of $\frac{\partial^2 F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i^2}$ at each one.

$$\begin{aligned}
\left.\frac{\partial^2 F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i^2}\right|_{\alpha_i=\alpha^{\pm}} &= -\frac{2}{(b-a)}\left(\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + x\right) + \frac{2(\alpha^{\pm}-a)}{(b-a)^2} \\
&\quad \left(\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + \left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=b} + 2x\right) && (4.103) \\
&= \pm\frac{2y}{(b-a)} && (4.104) \\
&< 0 \text{ for a maximum.} && (4.105)
\end{aligned}$$

Hence we must take the negative sign in 4.101. The cubic maxima α^- is therefore,

$$\alpha^{-} = a + (b-a)\left(\frac{\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + x - y}{\left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=a} + \left.\frac{\partial F(\lambda_i+\alpha_i d_i)}{\partial \alpha_i}\right|_{\alpha_i=b} + 2x}\right) \tag{4.106}$$

4.10.2 Interval Location

The purpose of this stage is to find an interval $[a, b]$ that satisfies 4.88 and either 4.89 or 4.90. The input to this stage is the starting point for the line search λ_i and the expected length of the interval.

If the first interval $[0, \text{length}]$ satisfies conditions 4.88 and 4.89 or 4.90, the stage returns. Otherwise a new interval is tested. Its left hand boundary comes from the old right hand boundary. The new right hand boundary is chosen to place the extrapolated cubic maxima in the center of the new interval. This extrapolated cubic maxima is subject to the restrictions that it must be to the right of the old interval and that the new interval length cannot grow by more than a factor of 8.

4.10.3 Interval Refinement

The purpose of this stage is to accurately locate the maxima of $F(\lambda_i + \alpha_i d_i)$ with respect to α_i. It does this by iteratively examining new points within the bracketing interval $[a, b]$ and uses the results to produce a new smaller interval. At all iterations the current interval satisfies 4.88 and 4.89 or 4.90, that is the interval is guaranteed to contain a maximum.

Trial points are chosen with cubic interpolation, with the restriction that trial points must lie between 20% and 80% into the interval. The search is terminated when a user specified tolerance or number of iterations is reached.

4.11 Summary

In this chapter we have obtained the ML, MMI and FD derivatives. We then described 5 derivative based optimisation algorithms, steepest ascent, conjugate gradients, quick-prop, on-line quick-prop and Manhattan optimisation. In addition, we described one non-derivative based optimisation algorithm, the heuristically extended Baum-Welch algorithm.

We can now proceed with an experimental evaluation of our new objective functions and optimisation algorithms. But first in the next chapter we shall again look at objective functions. This time we shall comment on a selection of objective functions which have been proposed in previous literature.

Chapter 5

Previous Literature

The study of HMM discriminative training has a large body of literature. As with all rapidly developing fields, the presentation is often disjoint and the conclusions contradictory. This section aims to review and align some of these studies.

5.1 Theoretical Comparisons

In [52] a differential learning strategy and a probabilistic learning strategy were compared. The task was classifying data produced from spherically symmetric normal distributions using a linear classifier with no bias. In all cases it was found that the probabilistic learning strategy performed best. This difference increased as the training set size decreased.

In [32–34, 36, 37], the authors present theoretical proofs and empirical tests illustrating the optimality of the CFM objective function. Unfortunately, the theoretical proof has a limited domain of applicability. It is valid when probabilities associated with distinct observations are unconnected. That is, there is no smoothness over feature space. Clearly in this case the attempt of a probabilistic objective function to match the per-observation probabilities is suboptimal due to the smoothness of the model. The experimental results in [32] were extremely encouraging. But the tests did involve under-parameterised models with large training sets. Conditions which are ideal for differential learning.

In [3] a simple 1-d two class normally distributed problem was studied. They show the CFM is optimal for training set error. They also provided a simple example where CFM could not distinguish between good and bad test set classifiers.

[74, 75] treats the case of classification by a model in an arbitrary mixture of domains. Each domain has its own priors and loss matrices. With non-uniform priors and loss matrices CFM is no longer optimal. Instead errors should be given different weights. Each domain in the mixture gives rise to different set of weights. Taking the value of the model as the expected value of the classification over all domains, results in a probabilistic objective function. Therefore it seems that the use of differentially trained acoustic models in most speech recognition systems is hard to justify because the use to which many acoustic models will be put is unknown.

In [29, 35, 53, 70], theoretical proofs were presented that showed a number of widely used objective functions do belong to the class of probabilistic objective functions. The functions examined were MMI, ML and minimum mean square error (MMSE). To a first order, this shows that the question of which probability density to model, is more important than which of these objective functions should be used.

In [39, 59] the authors derived the weightings given by ML, MMI and MMSE to training utterances. They showed that ML weighted all training examples equally. MMI gives a low weight to training examples which are correctly classified and far from the decision boundary. MMSE on the other hand, gives low weight to training examples which are correctly classified or misclassified and far from the decision boundary. This shows that for the incorrect model case (for the correct model case ML is optimal — see chapter 3), given enough training data, MMSE by focusing on marginal decisions, should exhibit lower training set error. The question of the model's value if the measure differs from error rate, was not tackled. Using the argument in [74, 75], it would seem to be problem dependent.

5.2 NN-HMM Hybrids

[56, 57, 68] presents the integration of a feed forward neural network (NN) with a Markov model. The NN in these works were trained to produce posterior probabilities of state occupancies. This was achieved by using the combination of a probabilistic objective function, conditional maximum likelihood, and Viterbi labelling of the training data. The results reported were very encouraging.

These systems are similar to our FD based systems apart from three differences.

1. These systems used a massive NN with explicit normalisation. Our system uses local PDFs and relies on the fact that normalisation is not required for classification. The global system contains more parameters and is difficult to implement for large vocabularies. Its traditional HMM counterpart, the semicontinous codebook [41, 42] is now rare. Another disadvantage of these systems is that the savings produced by beamwidth pruning is very much reduced. The entire NN must be active at all times in the systems.

2. These works utilised a two stage optimisation process, alignment followed by function fitting. On a purely computational viewpoint, this is suboptimal when on-line optimisation methods are used.

3. And finally, the HMM-NN hybrids presented in these works were trained with binary Viterbi labelling of the speech frames.

An elaboration of these works, known as REMAP, was presented in [38]. This work showed how the binary Viterbi segmentation could be replaced with a soft probabilistic segmentation. In computational terms however, their REMAP model training is not competitive when compared to FD. Even using a simplified time-invariant NN this algorithm is slower. It requires a forward-propagation during the probabilistic segmentation and a back-propagation when the derivatives are accumulated. Our current implementation of FD training requires only a forward propagation. The cost of derivative accumulation is made negligible by using pruning on the posterior probabilities (see page 18). The use of pruning techniques during the forward propagation, for example gaussian selection [9, 27, 64, 65], would increase this factor of two advantage. Our discriminative HMM systems also have the advantage in that they can be initialised with the Baum-Welch algorithm, further shortening the training time.

In [43, 44, 78] the authors presented a learning vector quantisation (LVQ)-HMM hybrid. LVQ is used instead of the more usual vector quantisation because the aim was to increase the codebook's accuracy of state classification. The latest work in [78], altered the basic model in an attempt to moderate the loss of information caused by binary classification in the codebook. In our system the closest match to this LVQ-HMM hybrid would be a differentially scored zero memory FD system. However, one may view the recognition process as an integration of local decision problems, and we have already mentioned that learning in a mixture of decision problems leads to a probabilistic objective function. The use of differential FD scoring can therefore be seen to be suboptimal.

In [72, 73] a recurrent NN was used as the interpolative probability estimating function. The hope is that the recurrent net can encode in its state units useful temporal information. This system roughly corresponds to a FD trained HMM utilising a N with some memory. One advantage of this recurrent NN approach is that the memory is discovered as part of the training process. There are however two disadvantages. Firstly the recurrent NN has to be run during recognition. Secondly if the assertions in this thesis are correct the use of contextual information does mean there is a danger that training data performance will be improved at the expense of generalisation. Whether this is important depends on the size of the training set and model flexibility.

5.3 HMM Studies

The use of "designer" differential learning for HMMs is currently very popular. In one of the earlier studies [51], the objective function was,

$$D' = \sum_{u=1}^{U} P_\lambda(w(u)|O(u)) \tag{5.1}$$

This objective function utilises the experimental observation that $P_\lambda(w(u)|O(u))$ is always close to one or zero, intermediate values are rarely seen. Under these conditions D' approximates the training set error.

[15] introduces different objective function D'', given by,

$$D'' = \sum_{u=1}^{U} \frac{1}{1+e^{-\gamma d(u)}} \tag{5.2}$$

where,

$$d(u) = P_\lambda(w(u)|O(u)) - \log\left(\frac{1}{N-1}\sum_{w \notin w(u)} P_\lambda(w|O(u))^\eta\right)^{\frac{1}{\eta}} \tag{5.3}$$

where,

η is a positive constant chosen to control the number of incorrect hypothesis which contribute to D''.

γ a positive constant which controls the smoothness of D''.

N is the total number of alternative transcriptions that can be recognised by the system.

Larger values of η and γ produce increasingly better approximations to training set error.

These objective functions unlike the one described in chapter 3 do not have any theoretical justification. Their design was guided by their creator's experience.

Both D' and D'' seek to make differential learning more robust by incorporating more confusion statistics. In one sense they can be considered as intermediate between differential learning and MMI. This does mean that D' and D'' lose the optimality properties of differential learning and MMI.

Like most other studies of "designer" differential learning [15, 51] reported improved results compared to a ML baseline. They also reported very much higher training set improvements compared to test-set improvements.

In [13, 60] the authors compared ML and MMI. Both studies presented encouraging results that showed MMI training could increase the performance of ML seed models. Both also showed that training performance improved more than test-set performance. The training set results in [60] were especially interesting. In [60], with the advantage of a better training algorithm, almost 100% training set accuracy was achieved. In this case the optima found by MMI coincided with one of the differential learning optima. Because of the probability distribution approximation property of MMI, which is not shared by differential learning, we would expect the test-set performance of MMI to be better than differential learning in that task.

Chapter 6

ISOLET Experiments

In this chapter we test our methods on the ISOLET database. The preprocessing, language model and acoustic model are standard ones that were not optimised in any way. Instead our independent variables are the objective function, optimisation algorithm, acoustic model complexity and test database. In section 6.4 we compare the speed of our optimisation algorithms against each other. The best ones were then used in later experiments. Sections 6.5 and 6.6 attempt to examine the properties of the MMI and FD objective function surfaces. Here we aim to verify a number of theoretical predictions. Perhaps the most important is that although MMI is provably asymptotically optimal for this realistic classification task, FD generalises better. Finally, in section 6.7 we present the conventional tables of results which show the effect of altering acoustic model complexity, test set and objective function.

6.1 ISOLET Database

The ISOLET speech database [18] contains 7800 examples of the English alphabet spoken in isolation. Two examples of each letter were recorded from 150 subjects.

The data was collected in the OGI speech recognition laboratory. The laboratory was 15' by 15' with a tile floor, standard office wall board and drop ceiling. In the room were two Sun workstations and three disk drives. At each prompt two seconds of speech was recorded with a Sennheiser HMD 224 noise cancelling microphone. It was then low-pass filtered at 7.6 kHz and 16 bit sampled at 16 kHz. Finally the uttered letter was isolated with a semi-automatic procedure and most of the surrounding silence was removed. The remaining recording was on average 0.5 seconds in length. The average signal to noise ratio was calculated as 31.5 dB.

The ISOLET experiments used the OGI suggested partitions. The training set consists of the first recording from 120 speakers. The multi-speaker test set consists of the second recording from the 120 training speakers, and the speaker-independent test set consists of both recordings from the remaining 30 speakers.

6.2 ISOLET Preprocessing

For the ISOLET database the standard Mel-frequency cepstral coefficients, MFCC, preprocessing as produced by HTK HCode V1.3 was used. Their production is illustrated in figure 6.1 and the steps are described in greater detail below:

1. The digitised speech is blocked by taking 25.6ms segments every 10ms. i.e. 409 samples at offsets of 160 samples.

2. Premphasis. Blocks are passed through first order filter

$$S'_n = \begin{cases} 0.0 & \text{if } n = 1, \\ S_n - S_{n-1} & \text{otherwise.} \end{cases} \tag{6.1}$$

   where
   S_n is the nth speech sample in the block.

3. A Hamming window is applied to the block. That is,

$$S''_n = 0.54 - 0.46 \cos\left(\frac{2\pi(n-1)}{408}\right) S'_n \quad \text{for } n = 1 \ldots 409 \tag{6.2}$$

4. The block is padded to 512 elements with trailing zeros. Then a discrete fourier transform is applied to produce 256 complex spectral domain values.

5. The magnitudes of the last 255 complex spectral values are binned (i.e. term zero, the direct current term is ignored). The bins are implemented as 24 triangular bandpass filters. The lower cutoff, middle knot and upper cutoff frequencies/indexes are given in table 6.1. Their spacing is chosen to approximate Mel-pitch scale which is given by,

$$g = 2595 \log_{10}(1 + f/700) \tag{6.3}$$

   where
   g is the frequency in Mels, and
   f is the frequency in Hertz.

6. The 24 binned values are logged.

7. The first 12 values of the discrete cosine transform are calculated.

$$[O(u)_t]_i = \sum_{j=1}^{24} m_j \cos\left(\frac{\pi i(j-0.5)}{24}\right) \quad \text{for } i = 1 \ldots 12 \tag{6.4}$$

   where
   m_j is the logged value of Mel-bin j.

8. A thirteenth element, the energy of the current frame $\sum_{n=1}^{409} S_n^2$ is appended.

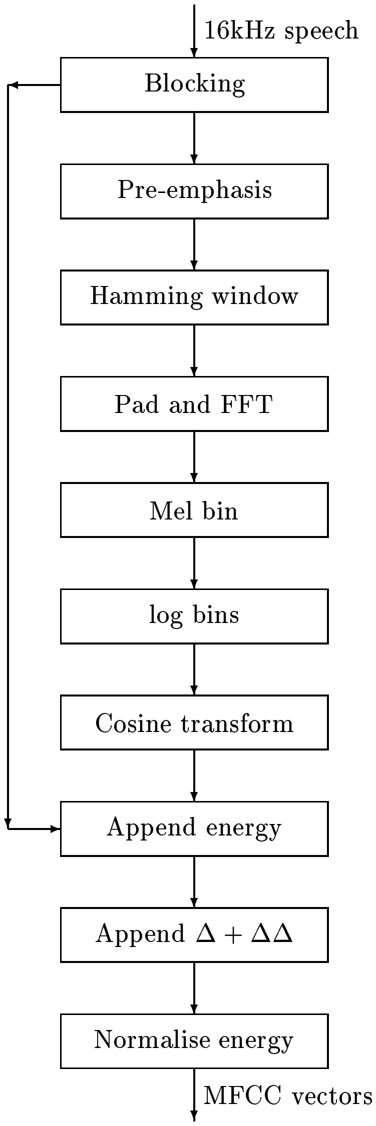

Figure 6.1: The ISOLET preprocessor

Bin	Lo-cutoff	Center	Hi-cutoff
1	0.000 Mel (0.000 Hz)	113.6 Mel (74.24 Hz)	227.2 Mel (156.4 Hz)
2	113.6 Mel (74.24 Hz)	227.2 Mel (156.4 Hz)	340.8 Mel (247.2 Hz)
3	227.2 Mel (156.4 Hz)	340.8 Mel (247.2 Hz)	454.4 Mel (347.6 Hz)
4	340.8 Mel (247.2 Hz)	454.4 Mel (347.6 Hz)	568.0 Mel (458.7 Hz)
5	454.4 Mel (347.6 Hz)	568.0 Mel (458.7 Hz)	681.6 Mel (581.6 Hz)
6	568.0 Mel (458.7 Hz)	681.6 Mel (581.6 Hz)	795.2 Mel (717.5 Hz)
7	681.6 Mel (581.6 Hz)	795.2 Mel (717.5 Hz)	908.8 Mel (867.9 Hz)
8	795.2 Mel (717.5 Hz)	908.8 Mel (867.9 Hz)	1022 Mel (1034 Hz)
9	908.8 Mel (867.9 Hz)	1022 Mel (1034 Hz)	1136 Mel (1218 Hz)
10	1022 Mel (1034 Hz)	1136 Mel (1218 Hz)	1250 Mel (1422 Hz)
11	1136 Mel (1218 Hz)	1250 Mel (1422 Hz)	1363 Mel (1647 Hz)
12	1250 Mel (1422 Hz)	1363 Mel (1647 Hz)	1477 Mel (1895 Hz)
13	1363 Mel (1647 Hz)	1477 Mel (1895 Hz)	1590 Mel (2171 Hz)
14	1477 Mel (1895 Hz)	1590 Mel (2171 Hz)	1704 Mel (2475 Hz)
15	1590 Mel (2171 Hz)	1704 Mel (2475 Hz)	1818 Mel (2812 Hz)
16	1704 Mel (2475 Hz)	1818 Mel (2812 Hz)	1931 Mel (3184 Hz)
17	1818 Mel (2812 Hz)	1931 Mel (3184 Hz)	2045 Mel (3596 Hz)
18	1931 Mel (3184 Hz)	2045 Mel (3596 Hz)	2158 Mel (4052 Hz)
19	2045 Mel (3596 Hz)	2158 Mel (4052 Hz)	2272 Mel (4556 Hz)
20	2158 Mel (4052 Hz)	2272 Mel (4556 Hz)	2386 Mel (5113 Hz)
21	2272 Mel (4556 Hz)	2386 Mel (5113 Hz)	2499 Mel (5730 Hz)
22	2386 Mel (5113 Hz)	2499 Mel (5730 Hz)	2613 Mel (6412 Hz)
23	2499 Mel (5730 Hz)	2613 Mel (6412 Hz)	2726 Mel (7166 Hz)
24	2613 Mel (6412 Hz)	2726 Mel (7166 Hz)	2840 Mel (8000 Hz)

Table 6.1: The HCode Mel-scale triangular bins

9. Then a further 26 elements are appended. These are the "delta" and "delta-delta" coefficients. The 13 "deltas" $[O(u)_t]_{14}, \ldots, [O(u)_t]_{26}$, approximate the rate of change of the base cepstral and energy coefficients, and are calculated as

$$[O(u)_t]_{i+13} = \begin{cases} [O(u)_{t+1}]_i - [O(u)_t]_i & \text{for } t = 1, 2 \\ \frac{\sum_{\tau=1}^{2} \tau([O(u)_{t+\tau}]_i - [O(u)_{t-\tau}]_i)}{2\sum_{\tau=1}^{2} \tau^2} & \text{for } t = 3, \ldots, T(u) - 2 \\ [O(u)_t]_i - [O(u)_{t-1}]_i & \text{for } t = T(u) - 1, T(u) \end{cases} \tag{6.5}$$

   The 13 "delta-deltas" $[O(u)_t]_{27}, \ldots, [O(u)_t]_{39}$, which approximate the acceleration of the base cepstral and energy coefficient are calculated by applying the above formula to the "deltas".

10. Each recording's log energy profile is scanned and low energy values are replaced with a floor 50dB's below the peak value. The energy profile is then scaled so the peak value is 1.0.

6.3 Previous Results

Extensive experiments on the ISOLET database have been published by the Oregon Graduate Institute [16–18]. In the database description [18], they report a 95% speaker-independent and a 96% speaker-dependent performance. In [16, 17] the recogniser is developed further and described in greater detail. The baseline classifier was a NN with 617 inputs, 52 hidden units and 26 output units. The input features to this NN were chosen using a combination of phonetic knowledge and experiment with the aim of yielding good classification performance. When trained on an extended training set consisting of the normal training and multi-speaker partitions this recogniser yielded a 95.9% performance. Adding a specialised e-set classifier as a post-processor for any initial e-set choice raised the augmented training set speaker-independent performance to 96%.

6.4 Comparison of Optimisation Methods

In this section we shall present an experimental comparison of the optimisation algorithms that were described in chapter 4. All experiments were carried out on the ISOLET database (see section 6.1), using our standard HMMs (see section 2.3), and using our standard ISOLET feature extraction (see section 6.2). The initial ML HMMs were trained using the Baum-Welch algorithm (see section 2.5) and all results were obtained on the speaker-independent test set.

When interpreting the results the reader should bear in mind a number of points.

1. None of our discriminative training algorithms guarantee monotonicity. Although the steepest ascent and conjugate gradient algorithms both use a line search, these algorithms are only monotonic if their line search is exact. This exactness requires their line search function evaluations to be close together in parameter space. This is not practical because of the computational cost. The author experimentally confirmed that when a (sometimes drastic), decrease in objective function occurred the algorithms managed to quickly recover.

2. Improvements measured by any one objective function does not imply improvements by any other measures. For instance, increasing likelihood score of a HMM does not necessarily increase its mutual information or FD score. If improvements were linked there would be little need for this work.

3. Training and test set improvements are not linked. Guaranteed linkage requires that the training and test sets are identical. But on most occasions we can expect that as the training set size increases (or equivalently our modelling flexibility decreases), the performance difference between training and test set will reduce.

The graphs in figure 6.2 plots the log conditional probability per frame $\frac{\sum_{u=1}^{U} \log P_\lambda(w(u)|O(u))}{\sum_{u=1}^{U} T(u)}$, or empirical word entropy, against epoch for each of our training algorithms.

The user settable parameters were chosen by some brief experimental trials in order to maximise the speed of convergence. These were set as follows,

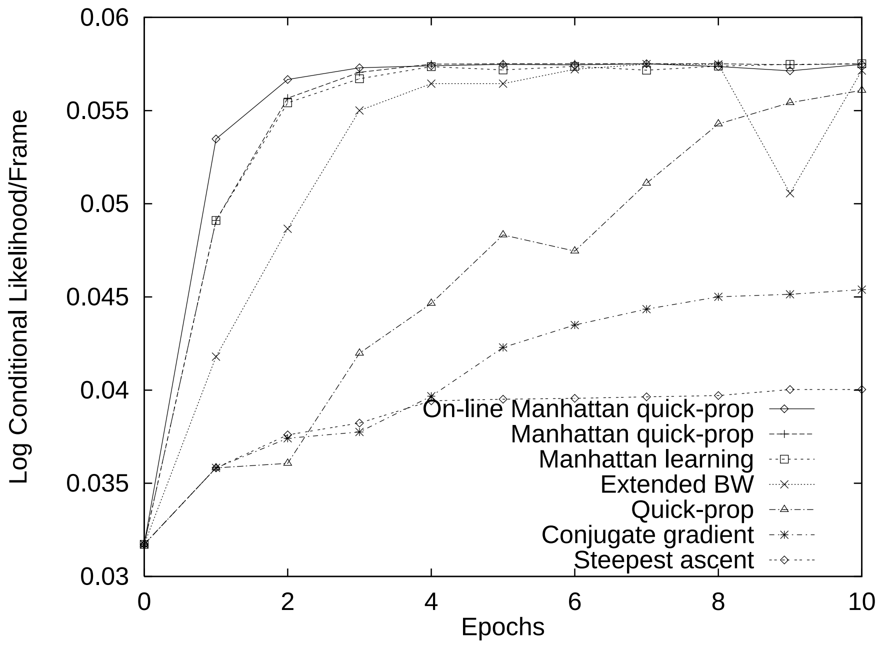

Figure 6.2: Word entropy v. epoch during MMI training for different algorithms

Steepest Ascent The line search was terminated when the step size α_k in equation 4.30 was found to a tolerance of 5%. This usually required around 6 function evaluations. The steepest ascent curve in figure 6.2 does not include this overhead. When comparing this curve with the curves for quick-prop, extended Baum-Welch, Manhattan learning, Manhattan quick-prop and on-line Manhattan quick-prop the reader should bear in mind that these latter algorithms do not have the hidden cost of a line search every epoch.

Conjugate Gradients Again the line search was terminated when the step size α_i in equation 4.50 was found to a tolerance of 5%. The plot in figure 6.2 assumes the line search cost is zero. This plot illustrates the Polak-Ribiere equation 4.47. The other update rules 4.46 and 4.48, have similar graphs.

Quick-prop The first update uses the steepest ascent rule. The following updates used quick-prop with a maximum growth rate G (in equation 4.54), of 1.5. Other growth rates were tried but the results were not fundamentally different.

Extended Baum-Welch There are no user settable parameters in our implementation.

Manhattan Learning The learning rate ϵ (in equations 4.70, 4.71, 4.72, 4.73 and 4.74), was set to 0.03 for the first iteration and 0.01 thereafter.

Manhattan quick-prop For the first update Manhattan learning was used. For later updates the quick-prop rule was used but a floor was imposed on the update size. This floor was set to the size of the Manhattan update with a global learning rate of 0.01.

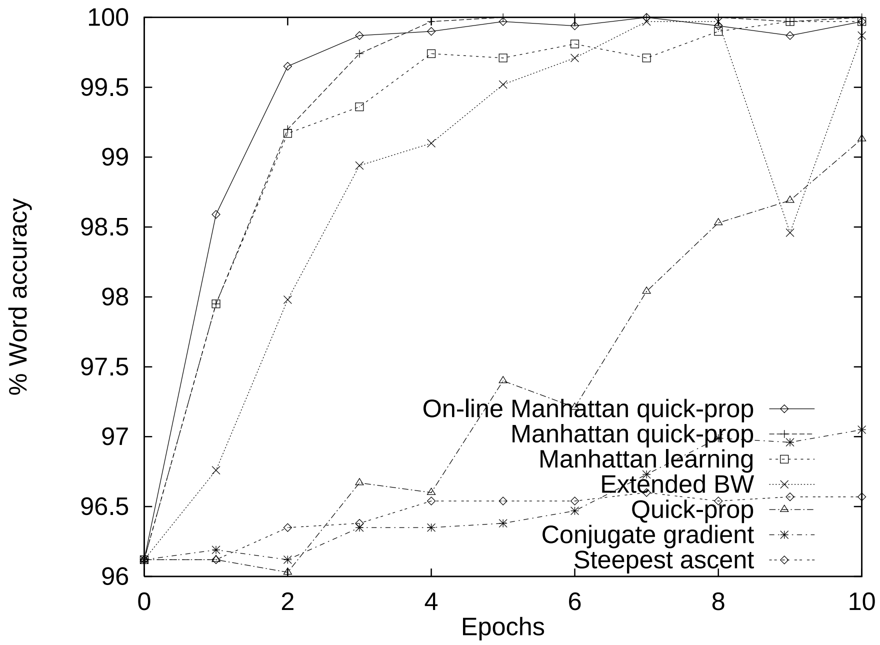

Figure 6.3: Word accuracy v. epoch during MMI training for different algorithms

On-line Manhattan quick-prop In our experiments we used 8 updates per epoch. The maximum growth rate was set to 1.1. The minimum step size was set to the Manhattan update with an initial global learning rate of 0.03 exponentially decaying to 0.01 at the end of the first epoch. For later updates it stayed at this floor value.

As can be seen in figure 6.2 the fastest algorithms are very close in speed. Figure 6.3 plots the training set accuracy against epoch. It again illustrates that there is very little to choose between the fastest algorithms. The fastest ones achieve 100% accuracy in 5 epochs. If we base our choice of training algorithm on final performance we cannot do any better. If we base our choice on speed of convergence there does not seem to be much room for improvement.

This conclusion changes when we examine algorithm convergence with the FD objective function. This is illustrated in figure 6.4 and 6.5. Figure 6.4 as before, plots the log conditional probability per frame and figure 6.5 plots the performance of the zero memory confusion HMM N. For the latter plot sequences of identical classifications were merged. These graphs show that the on-line Manhattan quick-prop algorithm is now better. One possible explanation for this effect is that during MMI training because there are very few training set errors, there is little training set redundancy to be exploited. On the other hand for the FD objective functions each training set partition provides a good estimate of the entire training set value and gradient. Notice that we do not achieve 100% performance. Because we do not know what the achievable accuracy is, it is possible that there are other algorithms that are substantially better.

Figures 6.4 and 6.5 also illustrate the remarkable robustness of our training algorithms. The Manhattan quick-prop algorithm fails at epoch 7, but manages by epoch 11 to fully recover.

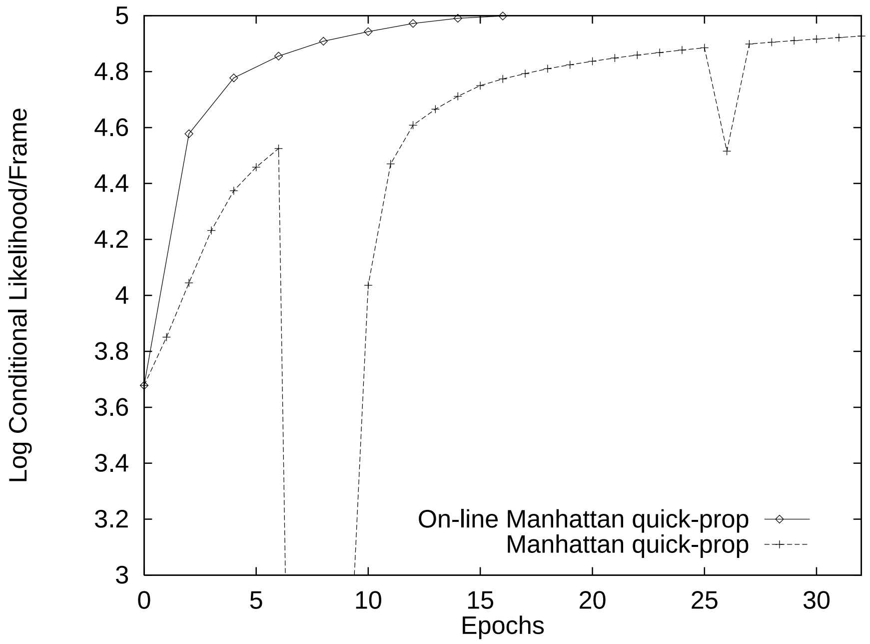

Figure 6.4: Frame entropy v. epoch during FD training for different algorithms

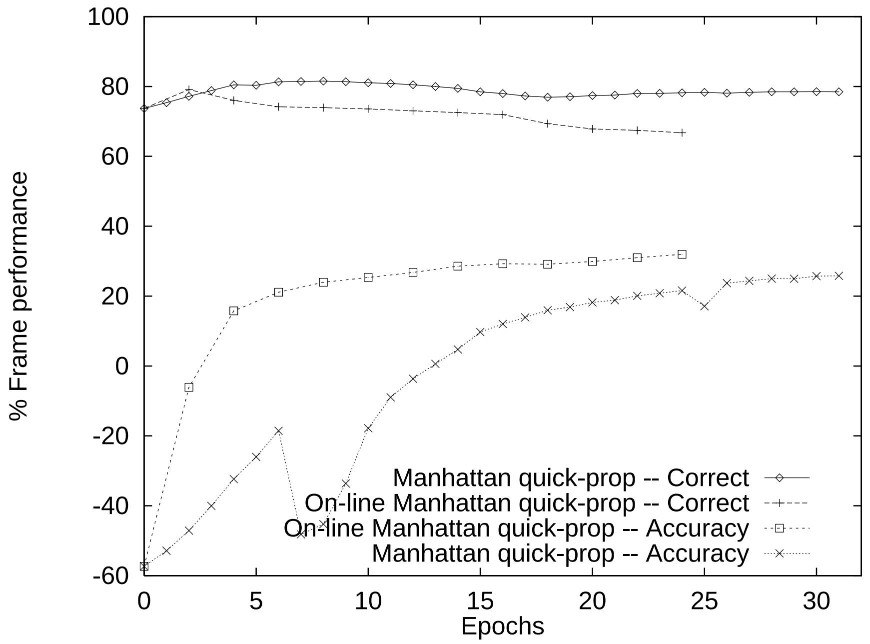

Figure 6.5: Frame accuracy v. epoch during FD training for different algorithms

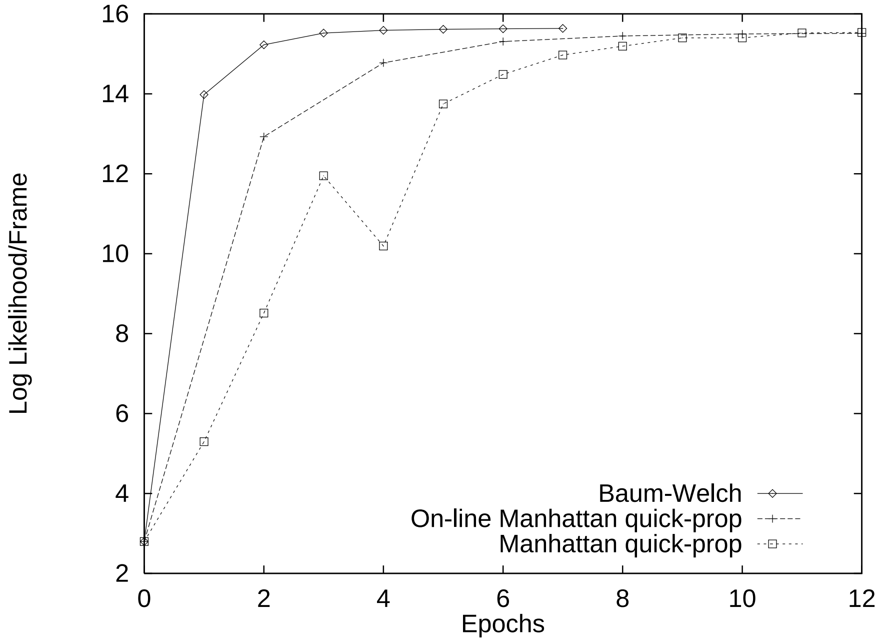

Figure 6.6: Log probability v. epoch during ML training for different algorithms

Figure 6.5 indicates that this behaviour may be due to the independence of the individual parameter updates. The relatively high classification performance during epoch 7 to 11 shows that many of the parameters have reasonable values. The training of these remain unaffected during the period in which the badly adjusted parameters recover. Although line-search procedures ensure that drastic reductions in an objective function are unlikely, the cost of line-search and the robustness of for instance, quick-prop, make their use uncompetitive.

To try and gauge the extent to which our training algorithms may be suboptimal we compared them to Baum-Welch (see section 2.5). Figure 6.6 plots normalised training set log probability against epoch for Baum-Welch, Manhattan quick-prop and on-line Manhattan quick-prop. The initial models were roughly initialised ML HMMs with all variances set to the same value (but not tied). As can be seen, on-line Manhattan quick-prop is superior to Manhattan quick-prop but slightly inferior to Baum-Welch. It is however comforting to see that the difference between on-line Manhattan quick-prop and Baum-Welch is small. In the author's opinion any improved discriminative training algorithm is not likely to substantially alter the conclusions reached in this dissertation. Figure 6.7 plots the training set accuracy against epoch for this experiment and shows the same trends. The drop in performance at epoch 1 for on-line Manhattan quick-prop and at epoch 4 for Manhattan quick-prop are an indicator that the likelihood metric lacks some vital information.

On the basis of these results all the FD experiments were done with the on-line Manhattan quick-prop algorithm and the MMI experiments were done with either Manhattan quick-prop or on-line Manhattan quick-prop.

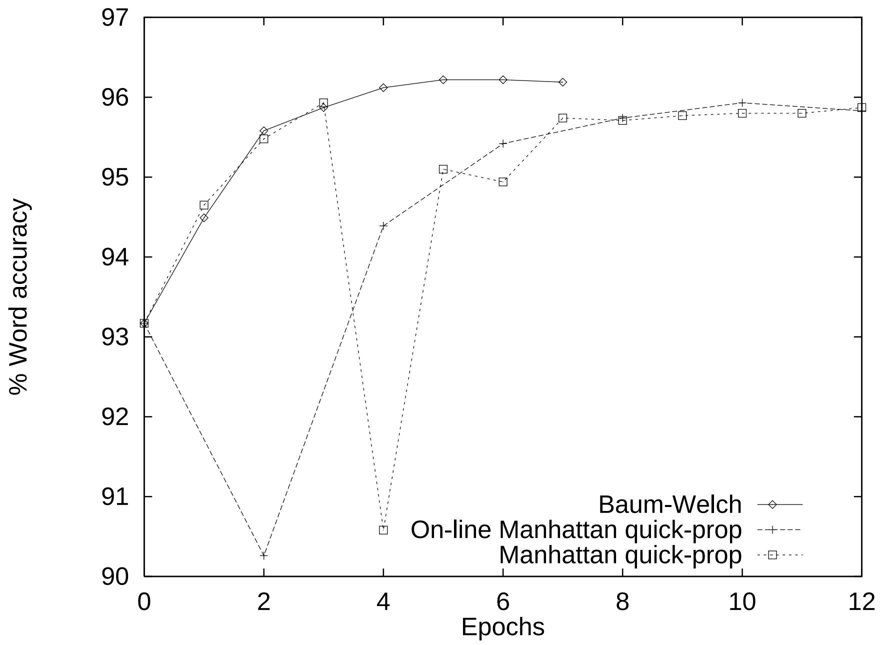

Figure 6.7: Word accuracy v. epoch during ML training for different algorithms

6.5 Temporal Evolution of MMI Training

Graphs 6.8 to 6.12 display the temporal evolution of various metrics during MMI training. The number of training epochs has been increased for the presentation in this section. In addition we used the on-line Manhattan quick-prop optimisation algorithm for training. These two factors ensure that an unusually wide area on the peak of the fitness surface will be sampled. Some features of note are,

1. In graphs 6.8 and 6.9 there is a large difference between the training set and speaker-independent test set improvements. In addition there is evidence of over-training as optimisation progresses.

2. In graph 6.10, the plot of normalised log probability against epoch, there gentle downward trend. It seems that there is enough flexibility even in a single mixture diagonal HMM for the model to maintain good likelihood and conditional likelihood values.

3. Graphs 6.11 and 6.12 plot the frame classification performance/entropy against epoch. We can see that as the word level entropy/accuracy increases, the frame entropy/accuracy decreases. The ML estimate, which scores the state PDFs independently of each other, produces a zero memory estimate of $P(s_t|O(u)_t)$. This makes the test set performance less dependent on the training set confusion coverage. As MMI training progresses the HMM parameters move away from their ML values and the frame performance suffers. The decreasing test set performance provides evidence that generalisation is the limiting factor.

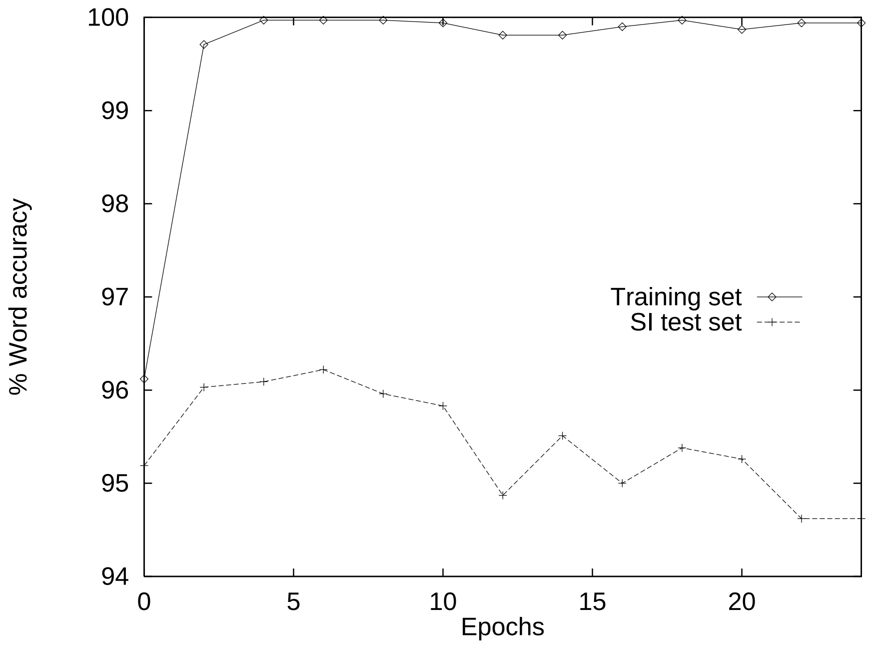

Figure 6.8: Temporal evolution of word accuracy during MMI training

6.6 Temporal Evolution of FD Training

Graphs 6.13 to 6.17 displays the temporal evolution of various metrics during FD training. Again we have attempted to sample a wide area on the peak of the fitness surface by using on-line Manhattan quick-prop and by increasing the number of training epochs. As we can see,

1. Graphs 6.13 and 6.14 plot the word level classification performance/entropy against epoch. There is an overall improvement as FD training progresses. In addition the difference between training and test set performances is reduced when compared to MMI training. The drop in performance in the middle of the experiment seems to be due to the natural variation between different performance measures. To verify that there is no over-training we repeated the experiment with a 8 mixture HMM. In every test the zero memory FD performance was above the initial ML performance. In order to test our conjecture that the generalisation performance of FD is the same as ML we attempted to sample the peak of the likelihood fitness surface. This however cannot be done using the Baum-Welch algorithm because of its convergence properties. Instead we applied on-line Manhattan quick-prop and ML objective function to our baseline single mixture HMMs. We did observe at the peak likelihood there was a performance variation of up-to 0.5%.

2. Graph 6.15 shows how the normalised log probability decreases as training progresses. Both distinct lines in this graph consist of two almost coincident curves, making a total of four curves. There is one curve for each possible pairing of either speaker independent test set/training set and transcription model $w(u)$/recognition model R. In all cases there

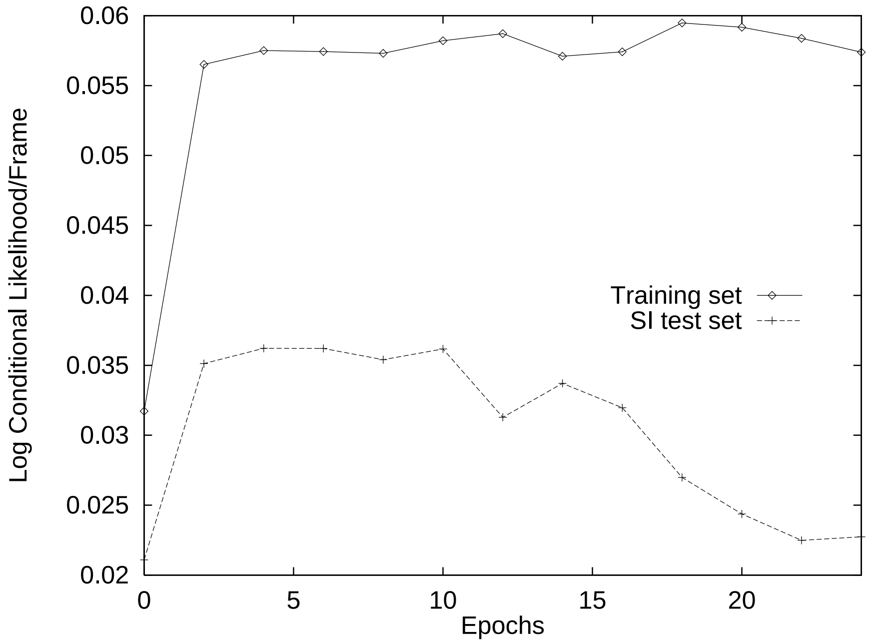

Figure 6.9: Temporal evolution of word entropy during MMI training

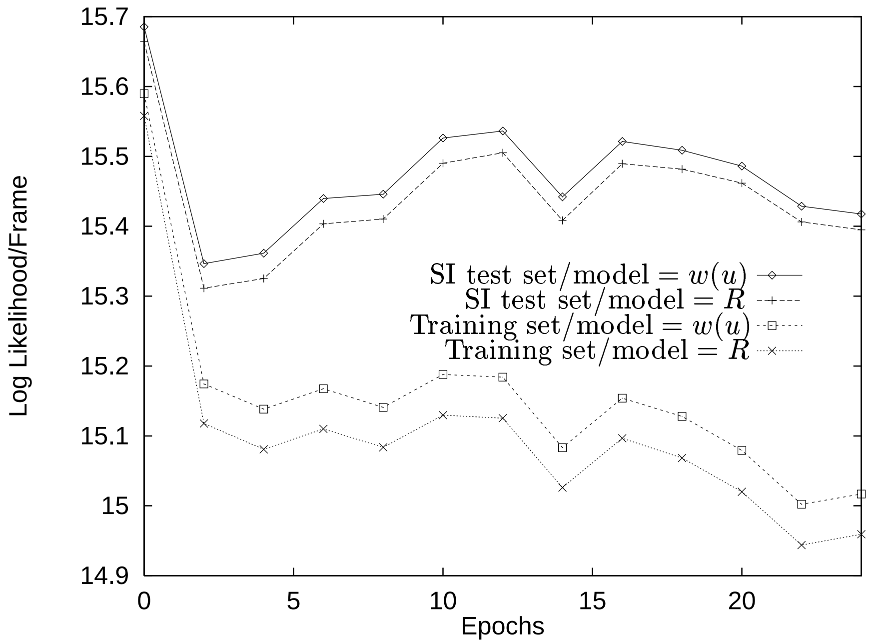

Figure 6.10: Temporal evolution of log probability during MMI training

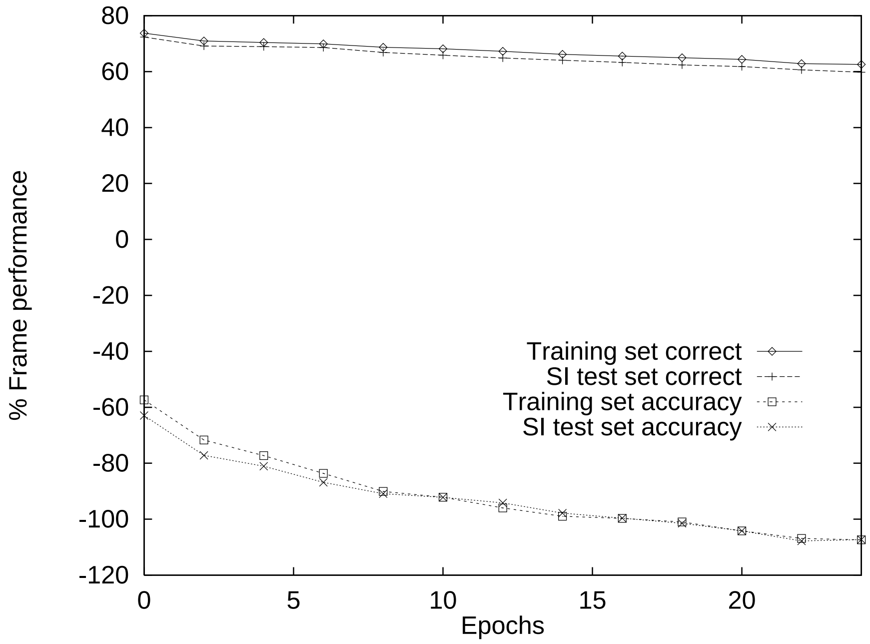

Figure 6.11: Temporal evolution of frame accuracy during MMI training

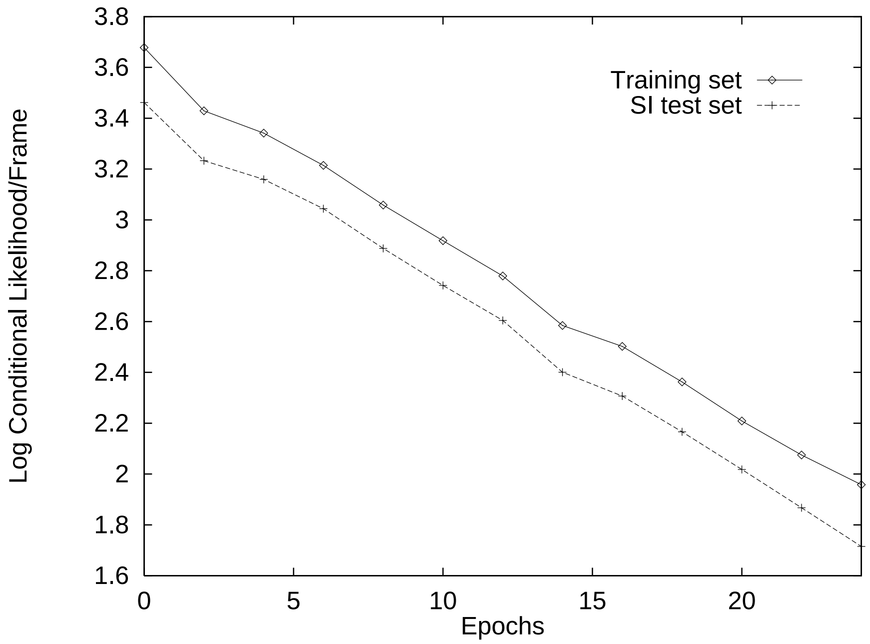

Figure 6.12: Temporal evolution of frame entropy during MMI training

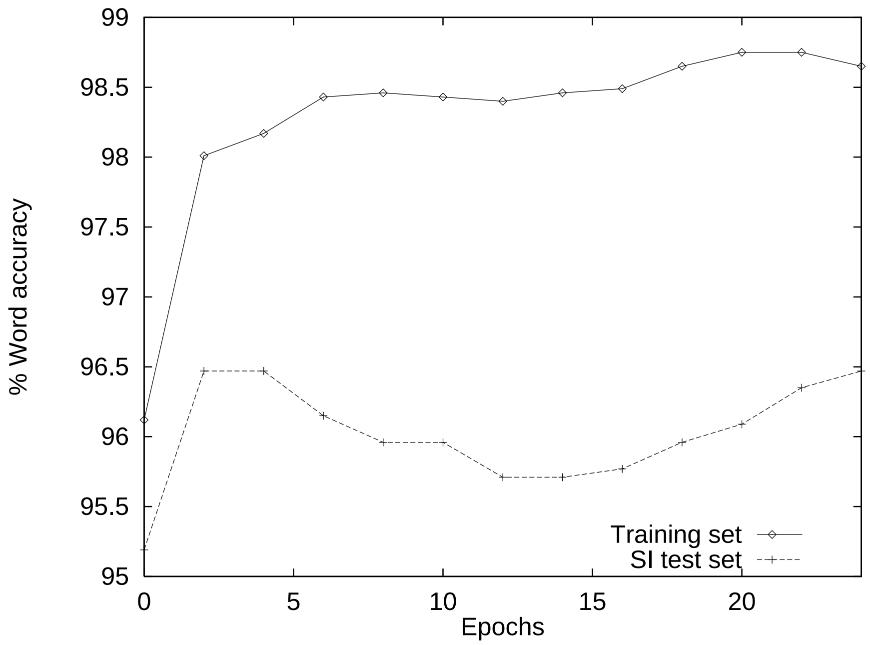

Figure 6.13: Temporal evolution of word accuracy during FD training

is a dramatic decrease in model likelihood. This does suggest that the parameters have to change substantially in order to improve the FD metric. This graph is a good illustration of how the likelihood metric is suboptimal for classification.

3. Graphs 6.16 and 6.17 plot the frame classification performance/entropy against epoch. As expected the frame recognition/entropy improves as FD training progresses.

6.7 ISOLET Results

Our ISOLET results are presented in tables 6.2, 6.3 and 6.4. The first column describes the type of HMM involved in the experiment. The globally tied diagonal variance HMM experiments are labelled "1 grand". The n diagonal mixture HMM experiments are labelled "n diag". And the single mixture full covariance experiments are labelled "1 full". The other column headings describe the source of the HMM and the objective function that is used. Columns labelled "ML" give our flat start ML results. Columns labelled "→MMI" give our ML seed, MMI training results. Columns labelled "→FD" give our ML seed, FD training results. And finally, the "→FD→MMI" column give our FD seed MMI training results.

Looking first at our ML results we can see two obvious trends. Increasing the model complexity increases the training set performance and up to the "16 diag" experiment, the speaker independent and multi-speaker results. Theory predicts this trend because increasing the HMM flexibility improves the probability predictions made by the HMMs. This improvement continues until there are too many parameters to estimate reliably. The second trend occurs between

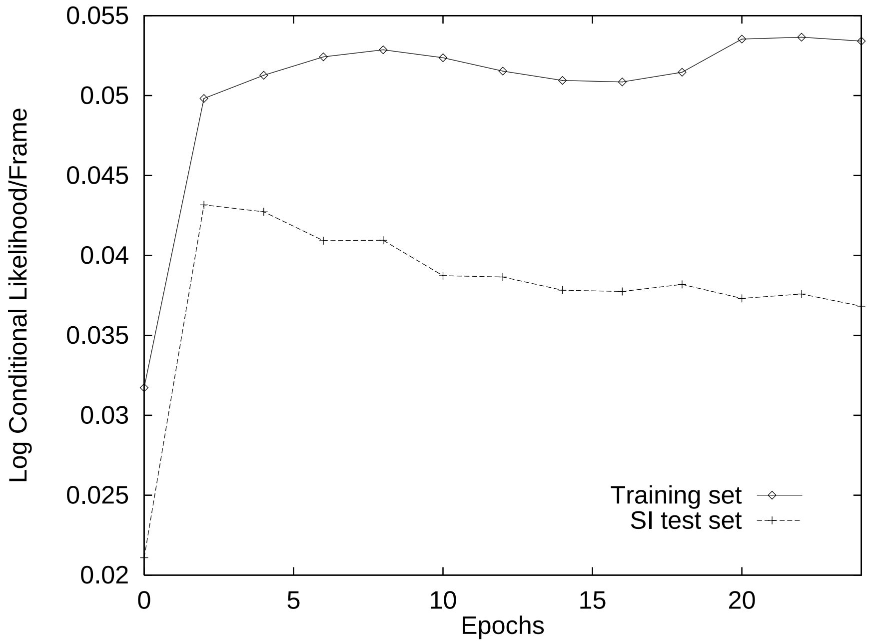

Figure 6.14: Temporal evolution of word entropy during FD training

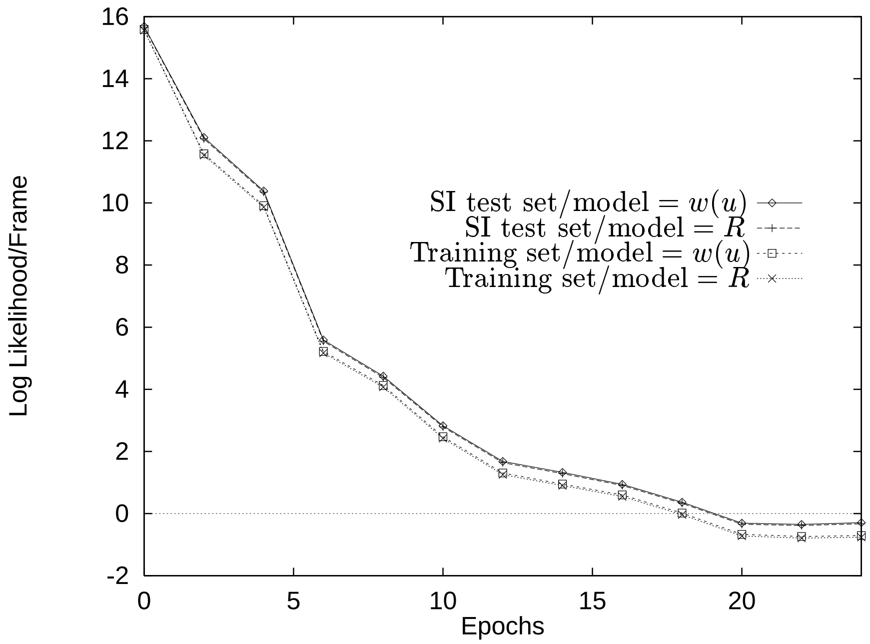

Figure 6.15: Temporal evolution of log probability during FD training

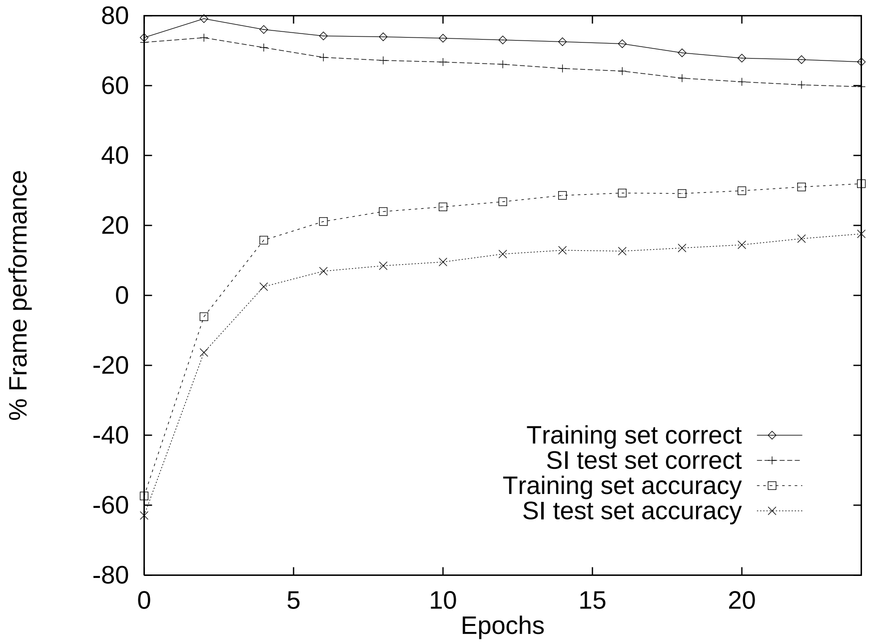

Figure 6.16: Temporal evolution of frame accuracy during FD training

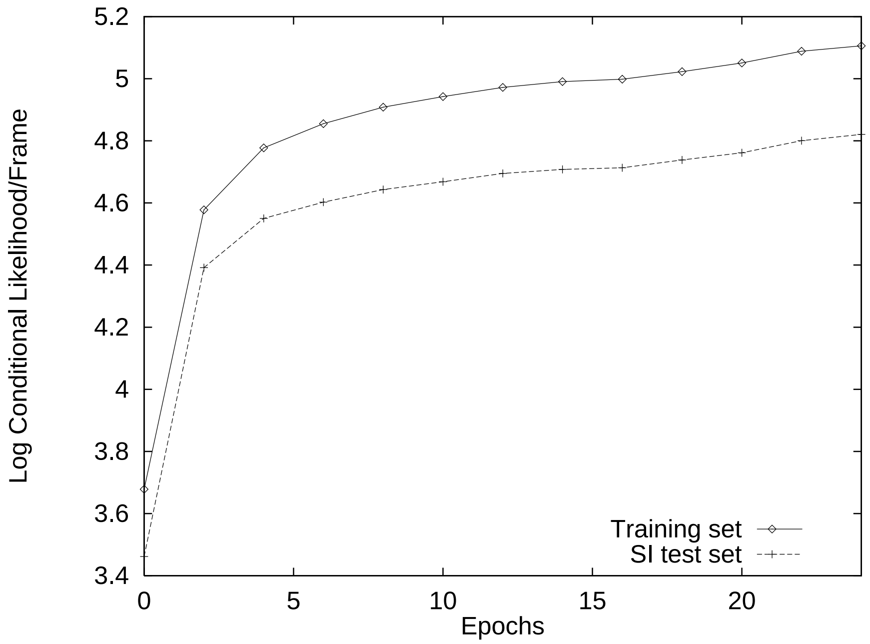

Figure 6.17: Temporal evolution of frame entropy during FD training

Type	Params.	ML	→MMI	→FD	→FD→MMI
1 grand.	8520	93.01%	95.26%	94.49%	96.35%
1 diag.	16827	95.19%	96.22%	96.47%	96.99%
2 diag.	33654	95.10%	96.21%	96.67%	96.67%
4 diag.	67308	95.96%	95.96%	96.54%	97.01%
8 diag.	134616	96.15%	96.47%	97.05%	*
16 diag.	269232	96.73%	*	*	*
32 diag.	538464	96.54%	*	*	*
1 full.	174660	96.99%	*	*	*

* Experiment was not deemed worthwhile because training set performance was too high.

Table 6.2: Speaker independent ISOLET results

Type	Params.	ML	→MMI	→FD	→FD→MMI
1 grand.	8520	93.17%	96.41%	95.22%	96.54%
1 diag.	16827	95.22%	96.41%	96.70%	97.24%
2 diag.	33654	96.25%	97.18%	97.44%	97.53%
4 diag.	67308	97.05%	97.56%	97.63%	97.82%
8 diag.	134616	97.60%	97.76%	97.95%	*
16 diag.	269232	97.88%	*	*	*
32 diag.	538464	97.85%	*	*	*
1 full.	174660	98.40%	*	*	*

* Experiment was not deemed worthwhile because training set performance was too high.

Table 6.3: Multi-speaker ISOLET results

Type	Params.	ML	→MMI	→FD	→FD→MMI
1 grand.	8520	94.74%	100%	96.51%	100%
1 diag.	16827	96.12%	100%	98.43%	100%
2 diag.	33654	97.40%	100%	99.20%	100%
4 diag.	67308	98.56%	100%	99.13%	100%
8 diag.	134616	99.20%	100%	99.78%	*
16 diag.	269232	99.33%	*	*	*
32 diag.	538464	99.45%	*	*	*
1 full.	174660	99.74%	*	*	*

* Experiment was not deemed worthwhile because training set performance was too high.

Table 6.4: Training set ISOLET results

different test sets. The training performance is always the highest followed by the multi-speaker performance and the lowest of all is the speaker independent performance. Again this trend can be predicted theoretically. There is an increasing mismatch from the training set to the multi-speaker and the speaker independent test sets.

We now examine the MMI experiments whose results are labelled "→MMI". The most striking aspect is the 100% training set accuracy. This implies that CFM or its empirical error variants would do no better than MMI. In fact we would expect that MMI would be better because of its attempt to model more from the data. For the speaker independent and multi-speaker test sets, MMI training does in all cases improve results. This occurs because MMI does not model $P_\lambda(O(u))$ and instead models the other aspects that are needed for classification. As expected from our theoretical discussion on page 29, this improvement decreases as the model complexity increases. When training with the larger models there were often large variations in performance from iteration to iteration and on occasions the performance would drop below the ML baseline.

Moving now to the "→FD" column we can see that in all cases these results were better than the ML ones. This again is predicted by theory. FD ignores the $P_\lambda(O(u)_t)$ values that are modelled by ML but not needed for classification. Comparing the results to MMI we see that for the "1 grand" experiment, MMI produced better HMMs. For the rest FD was better. This is a result of two effects. In the "1 grand" case the FD HMM does not have enough flexibility to model all the confusions introduced by the relaxed recognition model N. Given that many of these do not occur in real speech and they only serve to mask the ones that do, test set performance suffers. When trained with MMI, the HMMs successfully models all the true training set confusions. This results in better test set performance. Larger HMMs can adequately model the extra confusions introduced by FD, so for these models FD is better than MMI. During prolonged FD training the performance always remained above the ML baseline.

Finally, the last column labelled "→FD→MMI" applies the MMI objective function to the FD trained models. We hope in these experiments that any performance improvement engendered by FD training is maintained after removing the last training set confusions by MMI training. We are relying here on the fact that given enough model flexibility, MMI training does not substantially alter the HMM parameters (see graph 6.10). As we can see the improvement obtained for the smaller models is impressive. This provides some evidence that better accuracy could be obtained if our FD training algorithm could be improved. However, for the larger models the performance improvement is less impressive. In fact for these the performance does often drop below the FD baseline.

Chapter 7

CONNEX E-set Experiments

The experiments in this chapter were undertaken in order to provide an independent check of the results in chapter 6. The author attempted to avoid any sub-conscious bias by deliberately restricting the number of experiments. All of them, apart from a few preliminary ML tuning experiments are reported here.

7.1 CONNEX E-set Database

The British Telecom e-set database contains 2458 examples of the British English e-set {B, C, D, E, G, P, T, V}, spoken in isolation. There are approximately 3 examples of each letter from 104 different speakers. 54 speakers were males and 50 were females. Bad examples were discarded. This database is a subset of the larger British Telecom CONNEX database [76]. It is known that there are 115 faulty files in the whole CONNEX database. But we used the e-set database as is, without any corrections.

The data was collected in a soundproof booth. At each prompt two seconds of speech was recorded using a high-quality headset microphone. The speech was then 16 bit sampled at 20 kHz with a bandwidth of 100 Hz–8 kHz. The uttered letter was then isolated via a semi-automatic procedure and most of the surrounding silence was removed.

All CONNEX e-set experiments used the standard partitions. The speakers were divided into two disjoint groups. Each group contained roughly the same balance of ages and genders. The test group contained 1239 recordings and the training group contained 1219 recordings.

7.2 CONNEX E-set Preprocessing

The CONNEX e-set database was preprocessed to produce MFCC coefficients. However, their production differed from the scheme described in section 6.2.

The process used is illustrated in figure 7.1 and the steps are described in greater detail below:

1. A standard 27 channel filter-bank SRU-Bank [40], was applied to the digitised speech. The filters were approximately Mel spaced, and produced 8-bit log energy values at 10ms intervals.

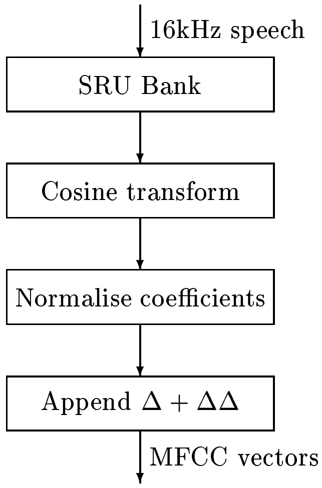

Figure 7.1: The CONNEX E-set preprocessor

2. 12 MFCC values were calculated by taking a discrete cosine transform. Unlike the OGI case, the zero'th coefficient was included.

$$[O(u)_t]_i = \sum_{j=1}^{27} m_j \cos\left(\frac{\pi(i-1)(j-0.5)}{27}\right) \quad \text{for } i = 1 \ldots 12 \tag{7.1}$$

   where
   m_j is the logged output of SRU-Bank channel j.

3. The MFCC coefficients were scaled, shifted and truncated so that each one, over the whole database, was an integer with a range 0 to 3123.

4. A further 24 elements were appended. These were the "delta" and the "delta-delta" coefficients, and were calculated using equation 6.5.

7.3 Previous Results

In [83] the authors achieved a baseline performance of 85.7%. The baseline HMM had 15 states with the last 9 states shared between all the models. The input features were the first 12 MFCCs and their differences. With the author's linear discriminant analysis algorithm the best performance the authors achieved was 92.1%. This HMM had the same topology and input features as their baseline model but used a subspace representation of a full covariance HMM. The 16 most discriminative dimensions of the input 24 dimensions were retained.

In [19] the authors reported results of experiments carried out on the full CONNEX database containing all 26 letters of the alphabet. Their HMM classifier had 15 states and 3 mixtures with a common diagonal covariance matrix per model. The input feature vector had 28 elements. The first 27 were the outputs of the SRUBank [40] preprocessor which were normalised by

Type	Params.	ML	→MMI	→FD	→FD→MMI
1 diag.	3081	85.32%	91.63%	91.96%	93.11%
2 diag.	6162	88.43%	92.45%	93.36%	93.77%
4 diag.	12324	91.80%	93.11%	94.34%	93.85%
8 diag.	24648	92.29%	93.44%	94.42%	94.59%
16 diag.	49296	93.03%	93.77%	94.28%	*
32 diag.	98592	92.86%	93.36%	94.40%	*
1 full.	31980	93.85%	93.93%	94.01%	*

* Experiment was not deemed worthwhile because training set performance was too high.

Table 7.1: Speaker independent CONNEX e-set results

Type	Params.	ML	→MMI	→FD	→FD→MMI
1 diag.	3081	90.15%	100%	95.96%	100%
2 diag.	6162	93.22%	100%	98.22%	100%
4 diag.	12324	95.56%	100%	99.27%	100%
8 diag.	24648	97.58%	100%	99.76%	100%
16 diag.	49296	98.14%	100%	99.84%	*
32 diag.	98592	98.31%	100%	100%	*
1 full.	31980	98.79%	100%	100%	*

* Experiment was not deemed worthwhile because training set performance was too high.

Table 7.2: Training set CONNEX e-set results

subtracting the average value from each component. The 28^{th} feature vector element was the frame energy. This classifier achieved 85.2% accuracy.

In [20] the authors presented the results of human recognition experiments on the full CONNEX database. After removing all faulty recordings tests gave a performance of 98.11%.

7.4 CONNEX E-set Results

Tables 7.1 and 7.2 give our CONNEX e-set results. These experiments were carried out in order to provide an independent confirmation of the results obtained in section 6.7 and used the recognition HMM illustrated in figure 2.3. Comparing tables 7.1 and 7.2 to tables 6.2 and 6.4 respectively, the reader can easily confirm that the same trends exist. The reader should read section 6.7 for an explanation for these trends.

Chapter 8

Conclusions

Most speech recognition research concentrates on finding better models for each stage of the speech recognition process. In the author's opinion this approach is correct — all stages can be improved. By listening to speech re-synthesised from preprocessed frames, we know that preprocessing discards useful information. Also, a HMM is not the true acoustic source. And, the prediction accuracy of our language models is far lower than a human's. As these shortcomings are reduced, speech recognition will become more robust and accurate.

However, in this dissertation we have adopted another approach. We have accepted a HMM as the acoustic model. We instead attempted to improve performance by better training. Section 8.1 reviews our work and summarises our conclusions. Section 8.2 suggest some of the more obvious avenues for further study.

8.1 Review of Work

Chapter 3 discussed the theoretical properties of various estimators. First we proved some widely known, large sample, exact model properties of ML estimators — sufficiency and efficiency. More importantly for our speech recognition task are the small sample, incorrect model properties. In this case ML estimators still enjoy the properties of invariance, probabilistic fitting and locality. Many authors have stated that ML estimation cannot be justified in the inexact model case. It is our contention that this view is wrong.

Chapter 3 continued with a description of three ML estimators. The first is known in speech literature as ML. The two others we described are MMI and FD. Training with speech literature ML produces HMMs which approximate $P^*(O(u)|w(u))$. The state PDFs in the resulting HMMs produce an estimate of zero memory $P^*(s_t|O(u)_t)$. That is, an estimate of $P^*(s_t|O(u)_t)$ where $O(u)_t$ is observed via a random zero memory process. This $P^*(s_t|O(u)_t)$ estimate is the key to understanding the excellent generalisation observed with ML. By ignoring the acoustic context of $O(u)_t$, ML extrapolates from the training data state confusions, to novel test data state confusions.

Training with MMI produces HMMs which approximate $P^*(w(u)|O(u))$. By not modelling $P^*(O(u))$ we reduce the "regressional stress" on the HMM. This increases training set performance. However test set generalisation suffers because this estimate does not extrapolate training set state confusions.

In this dissertation we concentrated on one member of the FD class of estimators, zero memory FD. HMMs trained with zero memory FD produce estimates of zero memory $P^*(s_t|O(u)_t)$. These HMMs have the same extrapolation ability as ML HMMs. In addition compared to ML, by modelling $P^*(s_t|O(u)_t)$ directly we ignore $P^*(O(u)_t)$, increasing the training set performance and effectively decreasing the training set noise.

Chapter 3 ended with a discussion about differential learning. This type of learning does not attempt to fit a probability distribution. By making strong assumptions about the use of the recogniser we can optimise application specific performance directly. The downside of course, is that the recogniser becomes application specific. For instance, CFM is optimal only for unweighted string error rate.

The optimisation of our objective functions is described in chapter 4. Here we obtained objective function derivatives and discussed how they could be used in a variety of optimisation algorithms. These optimisation algorithms were evaluated in chapter 6. We found that quickprop update of the hessian information; flooring of the step size with Manhattan learning and an on-line strategy all improved speed. In experiments reported further on, the vast majority of the best results were obtained in under 6 epochs.

Also in chapter 4 we sampled the peaks of our various objective functions. This empirically verified that zero memory FD generalised better that MMI. In addition, the experiments also supported the notion that zero memory FD generalised as well as ML.

Lastly, in the final part of chapter 6 and in chapter 7 we considered the effect of changing model complexity. In all cases, the peak MMI performance and all zero memory FD performances were better than ML. Although in the large training data set size limit, MMI should be best, in almost all cases zero memory FD was better. Previously, the poor generalisation of MMI HMMs was attributed to over-training. That is, training for too many epochs or using a model that had too many parameters. Here we have an experimental confirmation that a major cause of poor generalisation is the failure of MMI to reward the extrapolation of training set state confusions.

8.2 Future Work

The success of discriminative training relies on the speed and accuracy of the learning algorithm. There is a huge literature on optimisation and any one work can only evaluate a few of them. In particular, in this work we did not study the promising on-line learning techniques in any great depth.

Many HMM-NN hybrids use a pruning technique called posterior phone pruning. In theory this type of pruning can be used with any probabilistically trained HMM. It would be interesting to experimentally verify this hypothesis. With the use of zero memory FD, there is a hope that posterior phone pruning will become more effective. Compared to ML HMMs, this increased effectiveness could occur, because the noise distribution $P^*(O(u)_t)$ is not modelled. Compared to HMM-NN hybrids increased effectiveness could occur because of the possibility of using gaussian selection to reduce the cost of calculating $P^*(s_t|O(u)_t)$.

Current successful large vocabulary systems rely on speaker adaption to achieve high recognition accuracy. We have not attempted to study these methods.

Lastly, large vocabulary speech recognition systems use large numbers of states and gaussians. Contemporary HMM-NN hybrids provide proof that large vocabulary implementations of FD are possible. In addition, zero memory FD does enjoy some advantages. It can be implemented in a cache friendly manner and pruning techniques, for example gaussian selection can also used.

Bibliography

[1] J. Aczel and J. Pfanzagl. Remarks on the measurement of subjective probability and information. *Metrika*, 11(2):91–105, September 1966.

[2] Shunichi Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307, 1967.

[3] Etienne Barnard. Performance and generalization of the classification figure of merit criterion function. *IEEE Transactions of Neural Networks*, 2(2):322–325, 1991.

[4] Etienne Barnard. Optimization for training neural nets. *IEEE Transactions on Neural Networks*, 3(2):232–240, March 1992.

[5] Roberto Battiti. First- and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4:141–166, 1992.

[6] Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, pages 1–8, 1972.

[7] Leonard E. Baum and J. A. Eagon. An inequality with applications to statistical prediction for functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360–363, 1967.

[8] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[9] E. Bocchieri. Vector quantization for the efficient computation of continuous density likelihoods. In *ICASSP Proceedings*, volume 2, pages 692–695, 1993.

[10] Enrico L. Bocchieri and George R. Doddington. Frame-specific statistical features for speaker independent speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(4):755–764, 1986.

[11] H. Bourlard and C. J. Wellekens. Speech pattern discrimination and multilayer perceptrons. *Computer Speech and Language*, 3:1–19, 1989.

[12] Hervé Bourlard and Christian J. Wellekens. Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1–12, 1990.

[13] Peter F. Brown. *The Acoustic-Modelling Problem in Automatic Speech Recognition.* PhD thesis, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, August 1987.

[14] Régis Cardin, Yves Normandin, and Evelyne Millien. Inter-word coarticulation modelling and MMIE training for improved connected digit recognition. In *ICASSP Proceedings*, volume 2, pages 243–246, 1993.

[15] W. Chou, C.-H. Lee, and B.-H. Juang. Minimum error rate training of inter-word context dependent acoustic model units in speech recognition. In *International Conference on Spoken Language Processing*, pages 1367–1370, 1994.

[16] Mark Fanty & Ron Cole. Speaker-independent English alphabet recognition: Experiments with the e-set. In *International Conference on Spoken Language Processing*, volume 2, pages 1361–1364, November 1990.

[17] Mark Fanty & Ronald Cole. Spoken letter recognition. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 220–226, 1990.

[18] Ron Cole. The ISOLET spoken letter database. Technical Report CSE 90-004, Department of Computer Science and Engineering, Oregon Graduate Institute of Science & Technology, 19600 N. W. Von Neumann Drive, Beaverton, OR 97006. E-mail cole@cse.ogi.edu, March 1990.

[19] S. J. Cox and J. S. Bridle. Simultaneous speaker normalisation and utterance labelling using bayesian/neural net techniques. In *ICASSP Proceedings*, pages 161–164, 1990.

[20] S. J. Cox, P. W. Linford, K. O. Chichlowski, and R. D. Johnston. Performance of humans on a isolated word speech recognition task. In *Proceedings Institute of Acoustics*, volume 16, pages 23–30, 1994.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B (methodological)*, 39:1–38, 1977.

[22] Paul S. Dwyer. Some applications of matrix derivatives in multivariate analysis. *Journal of the American Statistical Association*, pages 607–625, June 1967.

[23] Scott E. Fahlman. An empirical study of learning speed in back-propagation. Technical Report CMU-CS-88-162, Carnegie Mellon University, September 1988.

[24] Marco Ferretti, Giulio Maltese, and Stefano Scarci. Language model and acoustic model information in probabilistic speech recognition. In *ICASSP Proceedings*, pages 707–710, 1989.

[25] P. Fischer. On the inequality $\sum p_i f(p_i) \geq \sum p_i f(q_i)$. *Metrika*, 18:199–208, 1972.

[26] Horacio Franco, Michael Cohen, Nelson Morgan, David Rumelhart, and Victor Abrash. Context-dependent connectionist probability estimation in a hybrid hidden Markov model-neural net speech recognition system. *Computer Speech and Language*, 8:211–222, 1994.

[27] J. Fritsch and I. Rogina. The bucket box intersection (BBI) algorithm for fast approximative evaluation of diagonal mixture gaussians. In *ICASSP Proceedings*, volume 2, pages 837–840, 1996.

[28] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[29] Herbert Gish. A probabilistic approach to the understanding and training of neural network classifiers. In *ICASSP Proceedings*, pages 1361–1364, 1990.

[30] P. S. Gopalakrishnan, D. Kanevsky, A. Nádas, and D. Nahamoo. A generalisation of the Baum algorithm to rational objective functions. In *ICASSP Proceedings*, pages 631–634, 1989.

[31] P. S. Gopalakrishnan, Dimitri Kanevsky, Arthur Nádas, and David Nahamoo. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):107–113, January 1991.

[32] J. B. Hampshire II. *A Differential Theory of Learning for Efficient Statistical Pattern Recognition*. PhD thesis, Carnegie Mellon University, 1993.

[33] J. B. Hampshire II and B. V. K. Vijaya Kumar. Shooting craps in search of an optimal strategy for training connectionist pattern classifiers. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 1125–1132. Morgan Kauffman, 1992.

[34] J. B. Hampshire II and B. V. K. Vijaya Kumar. Differential learning leads to efficient neural network classifiers. In *ICASSP Proceedings*, volume 1, pages 613–616, 1993.

[35] J. B. Hampshire II and B. A. Pearlmutter. Equivalence proofs for multilayer perceptron classifiers and the bayesian discriminant function. In Touretzky, Elman, Sejnowski, and Hinton, editors, *Proceeding of the 1990 Connectionist Models Summer School*, pages 159–172, 1991.

[36] John B. Hampshire II and B. V. K. Vijaya Kumar. Why error measures are sub-optimal for training neural network pattern classifiers. In *IEEE International Joint Conference on Neural Networks*, volume 4, pages 220–227, 1992.

[37] John B. Hampshire II and Alexander H. Waibel. A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Transactions on Neural Networks*, 1(2):216–228, 1990.

[38] Yochai Konig Herve Bourlard and Nelson Morgan. REMAP: Recursive estimation and maximisation of a-posteriori probabilities. Application to transition-based connectionist

speech recognition. Technical Report TR-94-064, International Computer Science Institute (ICSI), EECS Department, University of California, Berkeley, California, March 1995.

[39] M. M. Hochberg, Les T. Niles, J. T. Foote, and Harvey F. Silverman. Hidden Markov model/neural networks training techniques for connected alphadigit speech recognition. In *ICASSP Proceedings*, pages 109–112, 1991.

[40] J. N. Holmes. The JSRU channel vocoder. *Proc. IEE*, 127 Pt. F(1):53–60, 1980.

[41] X. D. Huang and M. A. Jack. Semi-continuous hidden Markov models for speech recognition. *Computer Speech and Language*, 3:239–251, 1989.

[42] X. D. Huang and M. A. Jack. Unified modeling of vector quantization and hidden Markov model using semi-continuous hidden Markov models. In *ICASSP Proceedings*, pages 639–642, 1989.

[43] H. Iwamida, S. Katagiri, and E. McDermott. Speaker-independent large vocabulary word recognition using an LVQ/HMM hybrid algorithm. In *ICASSP Proceedings*, pages 553–556, 1991.

[44] H. Iwamida, S. Katagiri, E. McDermott, and Y. Tohkura. A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *ICASSP Proceedings*, pages 489–492, 1990.

[45] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In E. L. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–387. New York: North-Holland, 1980.

[46] T. T. Jervis and W. J. Fitzgerald. Optimization schemes for neural networks. Technical Report CUED/F-INFENG/TR 144, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, October 1993.

[47] S. Kapadia, V. Valchev, and S. J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *ICASSP Proceedings*, volume 2, pages 491–494, 1993.

[48] D. J. Kershaw, M. M. Hochberg, and A. J. Robinson. Incorporating context-dependent classes in a hybrid recurrent network-HMM speech recognition system. Technical Report Cambridge/F-INFENG/TR217, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, 1995.

[49] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

[50] Louis A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734, September 1982.

[51] A. Ljolje, Y. Ephraim, and L. R. Rabiner. Estimation of hidden Markov model parameters by minimizing empirical error rate. In *ICASSP Proceedings*, pages 709–712, 1990.

[52] Ronny Meir. Empirical risk minimization versus maximum-likelihood estimation: A case study. *Neural Computation*, 7:144–157, 1995.

[53] John W. Miller, Rod Goodman, and Padhraic Smyth. On loss functions which minimise to conditional expected values and posterior probabilities. *IEEE Transactions on Information Theory*, 39(4):1404–1408, July 1993.

[54] John Moody and Joachim Utans. Principled architecture selection for neural networks: Application to corporate bond rating prediction. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 683–690, 1991.

[55] John E. Moody. The *effective* number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 847–854, 1991.

[56] N. Morgan and H. Bourlard. Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *ICASSP Proceedings*, pages 413–416, 1990.

[57] Nelson Morgan and Hervé A. Bourlard. Neural networks for statistical recognition of continuous speech. *Proceedings of the IEEE*, 83(5):741–770, May 1995.

[58] Arthur Nádas. Hidden Markov chains, the forward-backward algorithm, and initial statistics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(2):504–506, April 1983.

[59] Les T. Niles, Harvey F. Silverman, and Marcia A. Bush. Neural networks, maximum mutual information training, and maximum likelihood training. In *ICASSP Proceedings*, pages 493–496, 1990.

[60] Yves Normandin. *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem.* PhD thesis, Department of Electrical Engineering, McGill University, Montreal, March 1991.

[61] Yves Normandin, Régis Cardin, and Renato De Mori. High-performance connected digit recognition using maximum mutual information estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 2(2):299–311, April 1994.

[62] Yves Normandin, Roxane Lacouture, and Régis Cardin. MMIE training for large vocabulary continuous speech recognition. In *International Conference on Spoken Language Processing*, pages 1367–1370, 1994.

[63] Yves Normandin and Salvatore D. Morgera. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *ICASSP Proceedings*, pages 537–540, 1991.

[64] Stefan Ortmanns, Thorsten Firzlaff, and Hermann Ney. Fast likelihood computation methods for continuous mixture densities in large vocabulary speech recognition. In *ICASSP Proceedings*, volume 1, pages 139–142, 1996.

[65] M. Padmanabhan, L. R. Bahl, D. Nahamoo, and P. de Souza. Decision-tree based quantization of the feature space of a speech recogniser. In *ICASSP Proceedings*, volume 1, pages 147–150, 1996.

[66] Thomas W. Parsons. *Voice and Speech Processing.* McGraw-Hill Book Company, 1986.

[67] Douglas B. Paul. Algorithms for an optimal A^* search and linearizing the search in the stack decoder. In *ICASSP Proceedings*, pages 693–696, 1991.

[68] Steve Renals, Nelson Morgan, and Hervé Bourlard. Probability estimation by feed-forward networks in continuous speech recognition. In *IEEE Workshop on Neural Networks for Signal Processing*, October 1991.

[69] Steve Renals, Nelson Morgan, Hervé Bourlard, Michael Cohen, and Horacio Franco. Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174, 1994.

[70] Michael D. Richard and Richard P. Lippmann. Neural network classifiers estimate bayesian *a posteriori* probabilities. *Neural Computation*, 3:461–483, 1991.

[71] Tony Robinson and Frank Fallside. Phoneme recognition from the TIMIT database using recurrent error propagation networks. Technical Report CUED/F-INFENG/TR.42, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, March 1990.

[72] Tony Robinson, Mike Hochberg, and Steve Renals. IPA: Improved phone modelling with recurrent neural networks. In *ICASSP Proceedings*, volume 1, pages 37–40, 1994.

[73] Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent neutral networks in continuous speech recognition. In Chin-Hui Lee, Frank K. Soong, and Kuldip K. Paliwal, editors, *Automatic Speech and Speaker Recognition — Advanced Topics*, chapter 19. Kluwer Academic Publishers, 1995.

[74] David B. Rosen. Cross-entropy vs. squared error vs. misclassification: On the relationship among loss functions. Technical report, Center for Biomedical Modelling Research, University of Nevada, Reno, May 1994.

[75] David B. Rosen. Scoring the forecaster by mean resulting payoff of a distribution of decision problems. In G. Heidbreder, editor, *Maximum Entropy and Bayesian Methods. Proceedings of the Thirteenth International Workshop.* Kluwer, Dordrecht, The Netherlands, 1994.

[76] J. A. S. Salter. The RT5233 alphabetic database for the connex project. Technical Report Technical Report RT52/G231/89, BT Laboratories, BT Laboratories, Martlesham Heath, near Ipswich, England, 1989.

[77] Charles W. Therrin. *Decision Estimation and Classification.* John Wiley & Sons, 1989.

[78] Kari Torkkola. New ways to use LVQ-codebooks together with hidden Markov models. In *ICASSP Proceedings*, volume 1, pages 401–404, 1994.

[79] Joachim Utans and John Moody. Selecting neural network architectures via the prediction risk: Application to corporate bond rating prediction. In *First International Conference on Artificial Intelligence Applications on Wall Street*, 1991.

[80] A. R. Webb, David Lowe, and M. D. Bedworth. A comparison of nonlinear optimisation strategies for feed-forward adaptive layered networks. Memorandum 4157, Royal Signals and Radar Establishment, Procurement Executive, Ministry of Defence, RSRE Malvern, Worcs., July 1988.

[81] S. S. Wilks. *Mathematical Statistics.* New York, Wiley, 1961.

[82] P. C. Woodland and S. J. Young. Benchmark DARPA RM results with the HTK portable HMM toolkit. In *Proceedings DARPA Workshop*, September 1992.

[83] Philip C. Woodland and David R. Cole. Optimising hidden Markov models using discriminative output distributions. In *ICASSP Proceedings*, pages 545–548, 1991.

[84] S. J. Young. The general use of tying in phoneme-based HMM speech recognisers. In *ICASSP Proceedings*, volume 1, pages 569–572, 1992.