Chapter 4

Discriminative learning theory

Discriminative learning presents an alternative to the classical probabilistic interpretation of pattern recognition, which links a pattern classification task to a distribution estimation problem. Given an observation, the Bayes decision rule leads to the minimum misclassification probability when the true observation distribution is known, by selecting the model with the highest *a posteriori* probability. Unfortunately, the true form of the distribution is rarely known in classification problems and a parametric form is often assumed for computational tractability. Furthermore, the parameters of the assumed distribution have to be estimated from a limited amount of training data. These factors contribute to the sub-optimality of the distribution estimation strategy for classifier design and leads to the consideration of an alternative strategy that attempts only to discriminate between observations from different classes, rather than attempting to estimate the true distributions. The Bayes decision rule can still be applied if the models are used to implement discriminant functions rather than distribution estimators.

Discriminative learning has been researched for many decades for pattern classification purposes, but it is in conjunction with efficient methods for training artificial neural networks (ANNs) [82] that the most prominent research in this field has taken place. Most algorithms for training ANNs make use of supervised feedback of some error or reward (discriminative

measure) in computing derivatives for the weights of the ANN. In this chapter we focus on a discriminative approach that can be applied to the training of HMMs and in particular can be applied to the adaptation of HMMs as that is the prime reason for our interest in discriminative learning techniques.

4.1 Discriminative optimisation criteria

The most useful training or learning strategies include a criterion or function that in some way measures the quality of a particular model, given observations from the process that is modelled. Since the criterion determines subjectively the performance of any particular model, it prescribes which models will be considered "better" models and which will be considered "poorer" models. Usually, models are desired that maximise the criterion. The maximisation of a criterion, however, is generally obtained in closed-form solution only for the simplest criteria and models, e.g. a least squares linear estimation problem. Therefore, in addition to defining a criterion, a method for determining useful model parameters based on the criterion has to be established. The choice of optimisation function is thus influenced both by its intended purpose, e.g. increasing class separation or decreasing the misclassification rate, as well as by the ease with which it can be optimised.

Before we proceed to discuss discriminative criteria for optimisation, we first review the non-discriminative criterion that is most widely used for HMM training, namely the *maximum likelihood* criterion, $\arg\max_{\boldsymbol{\lambda}} f(\mathbf{X}|\boldsymbol{\lambda})$, of which the application was also discussed in Chapters 2 and 3. ML can be shown [83] to produce the best decoder if certain conditions are met, such as having a suitably large amount of data and knowing the true form of the distribution of the speech observations. Unfortunately, both conditions are not met for speech recognition problems. The ML criterion is functionally simple, but has the difficult goal of attempting to estimate the parameters of a distribution, rather than just a discriminant function. Even for ML estimation, closed form solutions do not exist for HMM parameter estimates and iterative methods such as the Baum-Welch procedure are

used to approximate the ML parameters. The Baum-Welch procedure guarantees increasing the ML criterion at each iteration until convergence in a locally optimal point, where discriminative optimisation techniques often deliver little in terms of guaranteeing increased performance and are prone to converge to (poor) local minima/maxima. ML model estimates are therefore almost always used as a starting point for discriminative training. After that, it is attempted to incrementally improve performance using discriminative optimisation techniques. We now proceed to discuss discriminative criteria applied to the training of HMMs.

4.1.1 Corrective training

Under the heading of corrective training we group various ad-hoc discriminative techniques that have been applied to the training of HMMs. Franco [84] suggested a heuristic discriminative training algorithm using a frame-level cost function to measure the degree of misalignment between a model and an observation sequence. The function measures at each time frame the squared error between the target function (set to one for the "correct state" of the true class models and zero otherwise) and the posterior state occupancy probability for the model at the time frame. The criterion function is thus a frame-level mean square error (MSE) function. Given an observation sequence, the method performs a gradient update on the parameters in the direction of a unity likelihood for the correct state of the correct model and in the direction of a likelihood of zero for any other state. The procedure theoretically stops when the correct state likelihoods are unity and the incorrect state (including all states from false class models) likelihoods are zero. Various heuristics were used to limit the degree of adaptation to avoid over-training. The method was shown to improve initial ML trained model performance on a phoneme recognition task.

Gauvain and Lee [58] proposed a simple modification to the statistics collection phase of the MAP algorithm to implement a heuristic version of corrective training. Training sentences that are incorrectly recognised are used as new data for updating model statistics and the state occupancy statistic, $\gamma_j(t)$ (Equation 2.11) is multiplied by -1 for an incorrect model

and by 1 for the correct model. A limited number of MAP adaptation iterations using the modified statistics are performed. The procedure has no explicit optimisation function, but was found to deliver better performance on connected digit and isolated word recognition tasks than ML estimated models.

Chen & Soong [85] suggested an N-best candidates-based discriminative training algorithm using a frame-level optimisation criterion. It improves on the method suggested by Franco [84] by not attempting to force zero-one state occupancy values for correct and incorrect classes respectively. The frame-level loss function comprises a half-wave rectified log-likelihood difference between the correct and competing hypotheses and is optimised on the training set by performing gradient descent on the HMM parameters. Performance improvement over ML trained HMMs was reported on connected digit and isolated word recognition tasks.

4.1.2 Maximum mutual information (MMI)

Maximum mutual information is an information-theoretic concept that provides a basis for the derivation of a discriminative training criterion. The following derivations closely follow McDermott [86]. The *conditional entropy* $H_{\Lambda}(\mathbf{C}|\mathbf{X})$ of the class random variable $\mathbf{C}$, given the observation random variable $\mathbf{X}$, is minimised in terms of the model parameters $\Lambda = (\lambda_1, ..., \lambda_M)$ when the *mutual information* is maximised for each of M classes. What this means is that the uncertainty associated with $\mathbf{C}$ given $\mathbf{X}$ is minimised when the model parameters Λ provide as much information as possible about the class random variable $\mathbf{C}$ given $\mathbf{X}$. This can be verified by noting that the mutual information between $\mathbf{C}$ and $\mathbf{X}$, $I_{\Lambda}(\mathbf{C};\mathbf{X}) = I_{\Lambda}(\mathbf{X};\mathbf{C})$ can be written as the difference between the entropy of $\mathbf{C}$ and the conditional entropy of $\mathbf{C}$ given $\mathbf{X}$:

$$I_{\Lambda}(\mathbf{C};\mathbf{X}) = H(\mathbf{C}) - H_{\Lambda}(\mathbf{C}|\mathbf{X}). \tag{4.1}$$

If the entropy $H(\mathbf{C})$, for a basic unit such as a word in a speech recognition task is expressed by a language model and may thus be considered a given [87] , minimisation of the conditional entropy is achieved by maximising the mutual information between $\mathbf{C}$ and $\mathbf{X}$.

A more useful form of Equation 4.1 will now be shown. With the entropy of a random variable $\mathbf{C}$ for model parameters $\mathbf{\Lambda}$ given by a summation over the class variable c

$$H(\mathbf{C}) = -\sum_{c} P(\mathbf{C}=c)\log P(\mathbf{C}=c) \tag{4.2}$$

and the conditional entropy of $\mathbf{C}$ given $\mathbf{X}$ for model parameters $\mathbf{\Lambda}$ given by a summation over c and the observation variable x by

$$H_{\mathbf{\Lambda}}(\mathbf{C}|\mathbf{X}) = -\sum_{c,x} P(\mathbf{C}=c,\mathbf{X}=x)\log P_{\mathbf{\Lambda}}(\mathbf{C}=c|\mathbf{X}=x), \tag{4.3}$$

it is convenient to examine the maximisation of Equation 4.1 expanded as follows [86]:

$$\begin{aligned}
I_{\mathbf{\Lambda}}(\mathbf{C};\mathbf{X}) &= -\sum_{c} P(\mathbf{C}=c)\log P(\mathbf{C}=c) + \sum_{c,x} P(\mathbf{C}=c,\mathbf{X}=x)\log P_{\mathbf{\Lambda}}(\mathbf{C}=c|\mathbf{X}=x) \\
&= -\sum_{c,x} P(\mathbf{C}=c,\mathbf{X}=x)\log P(\mathbf{C}=c) + \sum_{c,x} P(\mathbf{C}=c,\mathbf{X}=x)\log \frac{P_{\mathbf{\Lambda}}(\mathbf{C}=c,\mathbf{X}=x)}{P_{\mathbf{\Lambda}}(\mathbf{X}=x)} \\
&= \sum_{c,x} P(\mathbf{C}=c,\mathbf{X}=x)\log \frac{P_{\mathbf{\Lambda}}(\mathbf{C}=c,\mathbf{X}=x)}{P(\mathbf{C}=c)P_{\mathbf{\Lambda}}(\mathbf{X}=x)} \\
&= \sum_{c,x} P(\mathbf{C}=c,\mathbf{X}=x)\log \frac{P_{\mathbf{\Lambda}}(\mathbf{X}=x|\mathbf{C}=c)}{P_{\mathbf{\Lambda}}(\mathbf{X}=x)}.
\end{aligned} \tag{4.4}$$

Since the true likelihood $P(\mathbf{C}=c,\mathbf{X}=x)$ is unknown, training samples of $\mathbf{C}$ and $\mathbf{X}$ are assumed to be representative of the true distribution and the MMI criterion is maximised by the $\mathbf{\Lambda}$ that maximises

$$f_{\mathrm{MMI}}(\mathbf{\Lambda}) = \sum_{c,x} \log \frac{P_{\mathbf{\Lambda}}(\mathbf{X}=x|\mathbf{C}=c)}{\sum_{c'} P_{\mathbf{\Lambda}}(\mathbf{X}=x|\mathbf{C}=c')P(\mathbf{C}=c')}. \tag{4.5}$$

The MMI criterion differs from the ML criterion ($P_{\mathbf{\Lambda}}(\mathbf{X}=x|\mathbf{C}=c)$) since the MMI crite-

rion maximises the relative likelihood of the correct class, rather than simply maximising the absolute likelihood. This introduces discrimination into the training procedure. The correspondence between the MMI criterion and the *a posteriori* class probability is evident, but note that the likelihood functions associated with the model (Λ) that maximises the MMI criterion are of a discriminatory nature and do not implement density estimators.

Some applications of MMI are briefly mentioned. Cardin *et al.* [88] and also Normandin & Morgera [89] applied MMI estimation to the training of parameters of HMMs. Parameters were initialised with ML estimates and MMI was performed in an adaptive mode, with smoothing applied during the parameter update. Both studies showed improvement from ML trained models on the TI/NIST connected digit database. Kapadia *et al.* [90] achieved improved continuous phoneme recognition on the TIMIT [31] database using MMI estimation.

While the MMI criterion clearly improves on the ML criterion in terms of taking into account both the correct and incorrect model likelihoods, it still does not directly reflect the classification performance of a system. This topic is addressed in the next section.

4.1.3 Minimum error rate

The goal of a classifier is ultimately to achieve the minimum possible error rate, if equal cost is associated with each error. This minimum error rate is achieved with a Bayes classifier in which, for any observation, the discriminant function associated with the largest *a posteriori* probability has the largest value. The MMI criterion expresses the functional form of the *a posteriori* estimate, thereby increasing class separation, but does not expressly minimise the error rate association with the estimate. The most direct optimisation of the error rate can be achieved with a criterion that hard-limits the difference between the true class and the highest false class discriminant functions. Discontinuous criteria are hard to optimise, however, and therefore a continuous criterion that emulates the error rate should be considered. We discuss such a method that was implemented and extensively used in

this thesis in more detail in the following section.

4.2 Minimum classification error approach

The minimum classification error (MCE) approach suggested by Juang & Katagiri [91] provides a technique for designing a classifier that approaches the objective of minimum classification error more directly than the methods discussed so far in this section. This is achieved by providing a criterion that accurately reflects the error rate of a classifier, yet is continuous and thus differentiable, facilitating optimisation of the parameters of the classifier. The method in general does not lead to closed-form re-estimation solutions for parameters and is thus used in conjunction with a gradient-based optimisation scheme. We now proceed to discuss the criterion for optimisation used in MCE.

4.2.1 Optimisation criterion

The sample risk, represented by the number of misclassifications in the training set, is the simplest and most direct function representing the error rate. It is, however, a piece-wise constant function and thus very difficult to optimise numerically since its derivatives contain no information. MCE training attempts to overcome the difficulty of directly optimising the error rate of a classifier on a set of data by defining a smoothed version of the error rate for optimisation. There are two key problems that have to be addressed namely

- measuring the distance between a correct and multiple incorrect classes and
- measuring the loss associated with a classification.

Misclassification measure

The decision boundary for a two class classification problem, with classes C_1 and C_2, is easily described in terms of the *a posteriori* probabilities by $P(C_2|\mathbf{X}) = P(C_1|\mathbf{X})$. It is, however, not easily extended to provide a measure of the distance between the correct class and multiple incorrect classes. One way of defining such a *misclassification measure* for an observation $\mathbf{X}$ from class i in terms of the class conditional log-likelihood functions, using the notation $g_j(\mathbf{X};\mathbf{\Lambda}) = \log f(\mathbf{X}|\boldsymbol{\lambda}_j)$, where $f(\mathbf{X}|\boldsymbol{\lambda}_j)$ is the class conditional likelihood function for class j, is by [28]:

$$d_i(\mathbf{X};\mathbf{\Lambda}) = -g_i(\mathbf{X};\mathbf{\Lambda}) + \log\left[\frac{1}{M-1}\sum_{j,j\neq i}^{M} e^{g_j(\mathbf{X};\mathbf{\Lambda})\eta}\right]^{1/\eta}, \tag{4.6}$$

where η is a positive number. The *misclassification measure* is a continuous function of all the classifier parameters $\mathbf{\Lambda}$ and attempts to emulate the Bayes decision rule, i.e. that for an ith class utterance $\mathbf{X}$, $d_i(\mathbf{X};\mathbf{\Lambda}) < 0$ implies correct recognition and $d_i(\mathbf{X};\mathbf{\Lambda}) > 0$ implies incorrect recognition. The value of η controls the relative significance of false class likelihoods. When η is large the term in brackets approaches $\max_{j,j\neq i} g_j(\mathbf{X};\mathbf{\Lambda})$, which is exactly the Bayes decision rule. For smaller η, competing classes with relatively smaller likelihoods are also taken into account, thereby deviating from the Bayes decision rule in a well defined manner and creating a soft decision boundary.

The *averaging* of the incorrect classes in Equation 4.6 is perhaps easier to understand when it is expressed in terms of the class conditional likelihood functions

$$\left[\frac{1}{M-1}\sum_{j,j\neq i}^{M} f_j(\mathbf{X};\mathbf{\Lambda})^{\eta}\right]^{1/\eta}. \tag{4.7}$$

Note that the misclassification measure (Equation 4.6) therefore actually expresses the ratio of the incorrect to correct class likelihoods, just in the log domain. When working with HMMs this is sensible because the likelihood values have a very large range, making direct subtraction of likelihoods almost meaningless.

Loss function

The misclassification measure (Equation 4.6) improves on the MMI criterion (Equation 4.5) in the sense that it defines a threshold (in the region of zero) for misclassification, even though the threshold is soft. In order to achieve the goal of emulating the expected misclassification rate the misclassification measure is embedded in a smoothed zero-one function such as the sigmoid function. The resulting function is then called the *loss function* and is given by

$$l_i(\mathbf{X};\mathbf{\Lambda}) = \frac{1}{1+e^{-\gamma d_i(\mathbf{X};\mathbf{\Lambda})+\theta}} \tag{4.8}$$

with θ normally set to zero and $0 < \gamma \leq 1$. When $d_i(\mathbf{X};\mathbf{\Lambda})$ is much smaller than zero, which implies correct classification, virtually no loss is incurred, while a large value of $d_i(\mathbf{X};\mathbf{\Lambda})$ leads to a loss close to one.

The criterion for minimisation can then be defined for a given training data set consisting of O observation sequences $\mathbf{X}_1..\mathbf{X}_O$ from a total of M classes $\{C_1,...,C_M\}$ by the *empirical loss*

$$L(\mathbf{X}_1..\mathbf{X}_O,\mathbf{\Lambda}) = \sum_{o=1}^{O}\sum_{i=1}^{M} l_i(\mathbf{X}_o;\mathbf{\Lambda})\,1(\mathbf{X}_o \in C_i) \tag{4.9}$$

where $1(\varphi)$ is the indicator function, taking on the value 1 when φ is true and 0 when it is false. Use of the *expected loss* presents an alternative to using the empirical loss, but has the associated problem that since the true distributions are unknown, current distribution estimates must be used in an iterative procedure. This would, however, also imply that calculation of the expectation is dependent on the classifier parameters, further complicating the optimisation function, and thus we use the empirical loss as optimisation criterion.

4.2.2 Gradient descent optimisation

A simple solution to minimise the empirical loss defined in Equation 4.9 is to use the gradient descent technique with batch-mode parameter updates

$$\Lambda_{n+1} = \Lambda_n - \epsilon_n \sum_{o=1}^{O} \sum_{i=1}^{M} 1(\mathbf{X}_o \in C_i) \nabla l_i(\mathbf{X}_o; \Lambda)|_{\Lambda=\Lambda_n} \tag{4.10}$$

where ϵ_n is the update parameter in iteration n and is chosen to be a suitable decreasing function of n. Note that we calculate the gradient using all available samples, also termed a deterministic update, since this improves the estimate of the gradient. A block mode update may be computationally cheaper to perform, or even an on-line update can also be used for real-time purposes, but we have not further pursued these two options.

A problem with the gradient descent technique is that it is suitable only for unconstrained minimisation, while the parameters of an HMM have definite constraints. Chou *et al.* [92] suggested making use of parameter transformations that remove the constraints in the transformed parameter space and thus facilitate the use of gradient descent optimisation. Details of the parameter transformations are given in the next section, along with the application of the MCE approach for HMMs.

4.2.3 HMM parameter update

A procedure for applying MCE to the training of the parameters of continuous density HMMs was suggested by Chou *et al.* [92] under the name *segmental GPD* (generalised probabilistic descent). Use of the name GPD is derived from the original MCE paper [91] which proposed using *probabilistic descent*, i.e. minimising the *expected loss* rather than the *empirical loss*. For practical reasons, however, we optimise the empirical loss in implementations of the approach. A more detailed discussion of the application of MCE for HMM training was later published by Juang *et al.* [28] and forms the basis for the discussion in this section. We note, however, that previous publications [92, 28] did not take

into account false class derivatives, i.e. the derivative of the loss function with respect to competing classes. We therefore extend the derivations to include both true and false class derivatives as was suggested by Kwon & Un [93] for the special case of the discriminative state-weighted HMM. For completeness we also provide transition probability derivatives.

HMM likelihood functions

Given that HMMs have been selected as the framework for modelling speech features, the *class conditional log-likelihood function* $g_i(\mathbf{X};\mathbf{\Lambda})$, $i = 1,..,M$ takes the form

$$g_i(\mathbf{X};\mathbf{\Lambda}) = \log f_i(\mathbf{X};\mathbf{\Lambda}) = \log f(\mathbf{X}|\mathbf{A}^{(i)}, \{b_j^{(i)}\}_{j=1}^{N}) \tag{4.11}$$

where the superscript i denotes the parameter set associated with class i. The *segmental* training procedure uses the Viterbi state-aligned likelihood function, which calculates the likelihood of Equation 4.11 along the state sequence with the highest likelihood, producing the log-likelihood function given by

$$\begin{aligned} g_i(\mathbf{X};\mathbf{\Lambda}) &= \log\{\max_{\mathbf{q}} f_i(\mathbf{X},\mathbf{q};\mathbf{\Lambda})\} \\ &= \log f_i(\mathbf{X},\bar{\mathbf{q}};\mathbf{\Lambda}) \\ &= \sum_{t=1}^{T}[\log a_{\bar{q}_{t-1}\bar{q}_t}^{(i)} + \log b_{\bar{q}_t}^{(i)}(\mathbf{x}_t)] \end{aligned} \tag{4.12}$$

where $\bar{\mathbf{q}}$ is the sequence with the maximum likelihood. As discussed in Chapter 2.1.2, the observation density in each state is a Gaussian mixture distribution, given in extended form and diagonal covariance for model i by

$$b_j^{(i)}(\mathbf{x}_t) = \sum_{k=1}^{K} c_{jk}^{(i)} \mathcal{N}[\mathbf{x}_t, \boldsymbol{\mu}_{jk}^{(i)}, \mathbf{\Sigma}_{jk}^{(i)}] = \sum_{k=1}^{K} \frac{c_{jk}^{(i)}}{(2\pi)^{(D/2)} \prod_{l=1}^{D} \sigma_{jkl}^{(i)}} e^{-\frac{1}{2}\sum_{l=1}^{D}(\frac{x_{tl}-\mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}})^2} \tag{4.13}$$

Parameter transformations

It is desirable to maintain the original parameter constraints in the HMMs when adaptation takes place such as $\sum_{j=1}^{N} a_{ij} = 1$, $a_{ij} \geq 0$, $\sum_{k=1}^{K} c_{jk} = 1$, $c_{jk} \geq 0$ and $\sigma_{jkl} > 0$. In order for the problem to remain an unconstrained problem that is suitable for direct optimisation by gradient descent (Equation 4.10), a transformation of the parameters is necessary. A set of transformed parameters $\bar{a}_{ij}$, $\bar{c}_{jk}$, $\bar{\mu}_{jkl}$ and $\bar{\sigma}_{jkl}$ can be calculated that will maintain the constraints on the original parameters [28]:

1. $a_{ij} \rightarrow \bar{a}_{ij}$, where $a_{ij} = \frac{e^{\bar{a}_{ij}}}{\sum_{j'} e^{\bar{a}_{ij'}}}$
2. $c_{jk} \rightarrow \bar{c}_{jk}$, where $c_{jk} = \frac{e^{\bar{c}_{jk}}}{\sum_{k'} e^{\bar{c}_{jk'}}}$
3. $\mu_{jkl} \rightarrow \bar{\mu}_{jkl} = \frac{\mu_{jkl}}{\sigma_{jkl}}$
4. $\sigma_{jkl} \rightarrow \bar{\sigma}_{jkl} = \log \sigma_{jkl}$.

The reverse transformations of a_{ij} and c_{jk} ensure that the coefficients remain positive and maintain the property of summing to one. The transformation of μ_{jkl} normalises the relative magnitude of the mean in each dimension by the variance of the component in that dimension. In the author's experience, this transformation is very important because without it the derivative of the loss function with respect to the mean contains the precision term (the inverse of the variance), rendering the mean update stable only for very small values of the update parameter ϵ. With the transform in place, the loss function derivative with respect to the mean is proportional to the variance, rendering the mean update stable for a much wider range of values of the update parameter. This can be understood intuitively by considering that the output values of a Gaussian are less sensitive to changes in the mean value in a dimension for which the variance is large than for changes in the mean value in a dimension for which the variance is small. Finally, the transformation of σ_{jkl} maintains the constraint that $\sigma_{jkl} > 0$ and also greatly reduces the sensitivity of the update for small values of the variance- thereby also helping to render the update stable for a greater range of update parameters. The transformation of σ_{jkl} essentially implies that it

is updated multiplicatively, making the magnitude of the update relative to the magnitude of σ_{jkl}.

Gradient descent update equations

From Equation 4.10 the following gradient descent update equations are derived (similar to [28]) for the transformed parameters belonging to class i:

$$\bar{\mu}_{jkl}^{(i)}(n+1) = \bar{\mu}_{jkl}^{(i)}(n) - \epsilon_n \sum_{o=1}^{O}\sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c) \frac{\partial l_c(\mathbf{X}_o;\mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}}\Big|_{\mathbf{\Lambda}=\mathbf{\Lambda}_n} \tag{4.14}$$

$$\bar{\sigma}_{jkl}^{(i)}(n+1) = \bar{\sigma}_{jkl}^{(i)}(n) - \epsilon_n \sum_{o=1}^{O}\sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c) \frac{\partial l_c(\mathbf{X}_o;\mathbf{\Lambda})}{\partial \bar{\sigma}_{jkl}^{(i)}}\Big|_{\mathbf{\Lambda}=\mathbf{\Lambda}_n} \tag{4.15}$$

$$\bar{c}_{jk}^{(i)}(n+1) = \bar{c}_{jk}^{(i)}(n) - \epsilon_n \sum_{o=1}^{O}\sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c) \frac{\partial l_c(\mathbf{X}_o;\mathbf{\Lambda})}{\partial \bar{c}_{jk}^{(i)}}\Big|_{\mathbf{\Lambda}=\mathbf{\Lambda}_n} \tag{4.16}$$

$$\bar{a}_{jj'}^{(i)}(n+1) = \bar{a}_{jj'}^{(i)}(n) - \epsilon_n \sum_{o=1}^{O}\sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c) \frac{\partial l_c(\mathbf{X}_o;\mathbf{\Lambda})}{\partial \bar{a}_{jj'}^{(i)}}\Big|_{\mathbf{\Lambda}=\mathbf{\Lambda}_n}. \tag{4.17}$$

Calculation of derivatives

The derivatives of the loss with respect to the transformed parameters $\bar{\mu}_{jkl}$, $\bar{\sigma}_{jkl}$ and $\bar{c}_{jk}$, which appear in Equations 4.14 to 4.16, can be expanded as is now shown for the derivative of the mean (Equation 4.14) in the following equations [28]. First the derivative of the loss function (Equation 4.8) is expanded via the chain rule to include the misclassification

measure d (see also Equation 4.8)

$$\frac{\partial l_c(\mathbf{X}, \mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}} = \frac{\partial l_c(\mathbf{X}, \mathbf{\Lambda})}{\partial d_c(\mathbf{X}, \mathbf{\Lambda})} \frac{\partial d_c(\mathbf{X}, \mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}} \tag{4.18}$$

where

$$\frac{\partial l_c(\mathbf{X}, \mathbf{\Lambda})}{\partial d_c(\mathbf{X}, \mathbf{\Lambda})} = \gamma l_c(d_c(\mathbf{X}, \mathbf{\Lambda}))\big[1 - l_c(d_c(\mathbf{X}, \mathbf{\Lambda}))\big]. \tag{4.19}$$

Next the derivative of the misclassification measure (Equation 4.6) with respect to the transformed mean parameter is expanded, noting that it is dependent on whether $i = c$, i.e. whether the derivative is with respect to a true class or false class parameter (similar to [93])

$$\frac{\partial d_c(\mathbf{X}; \mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}} = -1(i = c)\frac{\partial g_i(\mathbf{X}; \mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}} + 1(i \neq c)\frac{e^{g_i(\mathbf{X};\mathbf{\Lambda})\eta}}{\sum_{f,f\neq c}^{M} e^{g_f(\mathbf{X};\mathbf{\Lambda})\eta}}\frac{\partial g_i(\mathbf{X}; \mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}}. \tag{4.20}$$

Finally the derivative of the class discriminant function (Equation 4.12) with respect to the transformed mean parameter is expanded to include the observation log-probability derivative

$$\frac{\partial g_i(\mathbf{X}; \mathbf{\Lambda})}{\partial \bar{\mu}_{jkl}^{(i)}} = \sum_{t=1}^{T(\mathbf{X})}\sum_{j=1}^{N} 1(\bar{q}_t = j)\frac{\partial \log b_j^{(i)}(\mathbf{x}_t)}{\partial \bar{\mu}_{jkl}^{(i)}}. \tag{4.21}$$

In Equations 4.18 through 4.21 we have now detailed the procedure for calculating the derivative of the loss function with respect to the observation log-probability. All that remains is to calculate the derivatives of the log-observation probability function $\log b_j^{(i)}(\mathbf{x}_t)$ (Equation 4.13) with respect to the transformed parameters $\bar{\mu}_{jkl}^{(i)}$, $\bar{c}_{jk}^{(i)}$ and $\bar{\sigma}_{jkl}^{(i)}$. The derivatives are given by [28]:

$$\frac{\partial \log b_j^{(i)}(\mathbf{x}_t)}{\partial \bar{\mu}_{jkl}^{(i)}} = \frac{c_{jk}^{(i)}\mathcal{N}[\mathbf{x}_t, \boldsymbol{\mu}_{jk}^{(i)}, \mathbf{\Sigma}_{jk}^{(i)}]}{b_j^{(i)}(\mathbf{x}_t)}\left(\frac{x_{tl}}{\sigma_{jkl}^{(i)}} - \bar{\mu}_{jkl}^{(i)}\right), \tag{4.22}$$

$$\frac{\partial \log b_j^{(i)}(\mathbf{x}_t)}{\partial \bar{\sigma}_{jkl}^{(i)}} = \frac{c_{jk}^{(i)} \mathcal{N}[\mathbf{x}_t, \boldsymbol{\mu}_{jk}^{(i)}, \boldsymbol{\Sigma}_{jk}^{(i)}]}{b_j^{(i)}(\mathbf{x}_t)} \left[\left(\frac{x_{tl}}{\sigma_{jkl}^{(i)}} - \bar{\mu}_{jkl}^{(i)} \right)^2 - 1 \right] \tag{4.23}$$

and we calculate

$$\frac{\partial \log b_j^{(i)}(\mathbf{x}_t)}{\partial \bar{c}_{jk}^{(i)}} = c_{jk}^{(i)} \left[\frac{\mathcal{N}[\mathbf{x}_t, \boldsymbol{\mu}_{jk}^{(i)}, \boldsymbol{\Sigma}_{jk}^{(i)}]}{b_j^{(i)}(\mathbf{x}_t)} - 1 \right]. \tag{4.24}$$

The derivative of the loss function with respect to the transformed transition probability parameter $\bar{a}_{ij}$, which appears in Equation 4.17 is also given in a similar form to Equations 4.18-4.20. Finally the derivative of the class discriminant function with respect to the transformed transition probability is given by

$$\frac{\partial g_i(\mathbf{X}; \boldsymbol{\Lambda})}{\partial \bar{a}_{jj'}^{(i)}} = \sum_{t=1}^{T(\mathbf{X})} \sum_{s=1}^{N} 1(\bar{q}_{t-1} = j)\, 1(\bar{q}_t = s)[1(j' = s) - a_{jj'}^{(i)}]. \tag{4.25}$$

Detailed derivations of Equations 4.24 and 4.25 are given in Appendix C because our equations differ from those previously published [86], where mixture weight and transition probability parameter dependencies were not taken into account.

4.2.4 MCE training for HMMs

With the update equations fully specified, the training procedure is now discussed in more detail. MCE training is usually preceded by standard ML training of models, such as expectation-maximisation (EM) training. EM training has the desirable property that it guarantees increased data-likelihood during training and is less prone to converge to local optima than gradient-based techniques. In contrast, training using MCE does not guarantee decreased loss (it depends on the selection of a suitably small update parameter) and is prone to converge to local minima.

The gradient descent optimisation approach discussed in this section, also termed segmental

GPD training, is implemented as follows:

1. ML models are estimated using the EM algorithm,
2. observation sequences are aligned with the models, accumulating the derivative statistics (Equations 4.18 through 4.25),
3. transformed parameters are updated (Equations 4.14 through 4.17), and
4. the reverse transformation of the parameters completes the process, which is iteratively repeated from point 2.

One of the problems with using gradient descent optimisation on MCE is that over-training may occur and that, based on the training set only, there are no suitable stopping criteria. Previous research reporting results using MCE/GPD used a fixed number of iterations and a linearly decreasing update parameter that was determined empirically to work well [28]. There are, however, also other parameters such as the slope of the sigmoid γ and the offset of the sigmoid θ that need to be carefully selected as they influence the stability, speed of convergence and ultimately the recognition performance achieved with the method. These issues are discussed in Chapter 5 when the method is applied to cross-language adaptation.

An alternative training method for MCE, that uses the N-best candidates from a search, was suggested by Chou *et al.* [94]. The method is also known as string-level MCE [95] and is particularly useful for optimising continuous speech recognition performance when a large amount of data is available. We discuss string-level MCE in detail in Section 4.5.1, where we compare it with the standard approach that was detailed in this section (also termed phoneme-level or model-level MCE) when extended using a cost-based method for improving word recognition performance.

4.2.5 Applications

One of the first applications of the MCE approach for speech recognition used artificial neural networks to classify features in isolated word speech experiments [91]. Other early research on the application of MCE for speech recognition investigated improving the performance of dynamic time warping-based systems. The MCE criterion was employed for the discriminative optimisation of several parameters of dynamic time warping (DTW) systems, including trajectory weighting coefficients [96] and reference patterns [97, 98]. MCE was also applied to both DTW [99, 100] and HMM-based [101] word-spotting systems, as well as for utterance rejection [102]. MCE was applied for the optimisation of standard feature extraction parameters in speech recognition [103, 104] as well as for optimising dynamic (trajectory) features [105]. The unified framework that MCE provides for global optimisation of both the feature extraction front end, as well as the classification back end of a system was also researched [106, 107].

Applications that benefited from the use of MCE in the above-mentioned papers include vowel recognition, the E-set problem and connected digit recognition. Relatively few studies have targeted improving performance on continuous speech. An N-best-based MCE optimisation approach was shown to improve continuous speech recognition performance for the DARPA naval resource management (RM) task [94] compared to ML trained models. Another study [95] found string-level MCE to improve continuous phoneme recognition performance compared to ML trained models but found phoneme-level MCE to outperform string-level MCE for the specific task.

Adaptation using MCE

MCE has recently been applied specifically for the purpose of adapting pre-trained models to better fit new speech data. Matsui & Furui [29] compared the MAP and MCE techniques for adaptation of Gaussian mean and mixture weight parameters and found that the best results were obtained for a combination of the MAP and MCE methods. MAP was used to

find an initial estimate of the speaker adapted parameters and MCE was used to further fine-tune the results. The reason for first using MAP is that MCE is prone to converging to local minima and therefore achieved better results when starting with the improved MAP estimated models rather than with the speaker independent models. McDermott *et al.* [108] applied MCE to on-line adaptation and found it to outperform a segmental k-means approach. Laurila *et al.* [109] performed adaptation of only Gaussian mean parameters of HMMs to new speakers and environmental noise using MCE, MMI, MLLR and MAP methods. They report that MAP and MCE delivered very similar results and produced better recognition performance than the MMI and MLLR approaches.

In this section we detailed the basic approach to MCE optimisation of HMM parameters. In the following sections we propose a few specific extensions to MCE. We propose extending MCE to discriminatively adapt duration modelling parameters since it is expected that all parameters, including duration modelling parameters, may need to be optimised for a new language. A method for the discriminative optimisation of linear parameter transformations is proposed that may deliver better performance than ML estimated transformations. Finally, we propose a method to modify the MCE misclassification measure in order to associate a (language specific) cost with misclassifying a class as a certain other class. This enables MCE to focus on the adaptation of class boundaries that are important for recognition in the target language.

4.3 Discriminative optimisation of duration modelling parameters

Performance improvement may be obtained by the discriminative optimisation of the duration modelling parameters in addition to the discriminative optimisation of the HMM parameters described in Section 4.2. In Section 2.1.3 we detailed the approach to explicit duration modelling followed in this thesis. A gamma distribution function is used to model the distribution of the number of frames spent in each frame of the HMM. The parameters

of the gamma distributions, namely the α and β parameters, are simply estimated through their relationship to the expected mean and variance of the number of frames spent in each state. Use of this estimation procedure has been substantiated empirically, but it is not guaranteed to deliver optimal performance, especially since the true form of the duration p.d.f. is unknown.

In this section we therefore propose the discriminative optimisation of duration modelling parameters using the MCE framework and derive the equations for it. The state aligned HMM likelihood function in Equation 4.12 can easily be expanded to include explicit duration modelling and is then given by

$$g_i(\mathbf{X};\mathbf{\Lambda}) = +\sum_{t=1}^{T}[\log a^{(i)}_{\tilde{q}_{t-1}\tilde{q}_t} + \log b^{(i)}_{\tilde{q}_t}(\mathbf{x}_t)] + \sum_{j=1}^{N}\log \rho_j(\tau_j) \tag{4.26}$$

where τ_j is the number of discrete time frames spent in state j and $\log \rho_j()$ is the duration log-likelihood function in state j given by (refer to Equation 2.4)

$$\log \rho_j(\tau_j) = \alpha_j \log \beta_j - \log \Gamma(\alpha_j) + (\alpha_j - 1)\log \tau_j - \beta_j \tau_j. \tag{4.27}$$

We note that a model duration likelihood function can also be used in conjunction with the state duration likelihood function, but we have not incorporated a model duration likelihood function. Also the transition probability parameters can be left out if one considers them to be replaced by explicit duration modelling.

Considering the duration modelling parameters part of the HMM modelling parameter set Λ, gradient descent optimisation of the duration modelling parameters α and β in state j of model i is implemented by the update equations

$$\alpha_j^{(i)}(n+1) = \alpha_j^{(i)}(n) - \epsilon_n \sum_{o=1}^{O}\sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c)\frac{\partial l_c(\mathbf{X}_o;\mathbf{\Lambda})}{\partial \alpha_j^{(i)}}\Big|_{\mathbf{\Lambda}=\mathbf{\Lambda}_n} \tag{4.28}$$

and

$$\beta_j^{(i)}(n+1) = \beta_j^{(i)}(n) - \epsilon_n \sum_{o=1}^{O}\sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c) \frac{\partial l_c(\mathbf{X}_o;\mathbf{\Lambda})}{\partial \beta_j^{(i)}}\Big|_{\mathbf{\Lambda}=\mathbf{\Lambda}_n}. \tag{4.29}$$

The partial derivative of the loss function with respect to the class discriminant function is given by (refer to Equations 4.18-4.21)

$$\frac{\partial l_c(\mathbf{X},\mathbf{\Lambda})}{\partial g_i(\mathbf{X};\mathbf{\Lambda})} = \gamma l_c(d_c)\big[1 - l_c(d_c)\big]\Big[-1(i=c) + 1(i\neq c)\frac{e^{g_i(\mathbf{X};\mathbf{\Lambda})\eta}}{\sum_{f,f\neq c}^{M} e^{g_f(\mathbf{X};\mathbf{\Lambda})\eta}}\Big]. \tag{4.30}$$

The partial derivatives of the class discriminant function (Equation 4.26) with respect to the parameters α and β are given by

$$\frac{\partial g_i(\mathbf{X};\mathbf{\Lambda})}{\partial \alpha_j^{(i)}} = \log \beta_j^{(i)} - \frac{\Gamma'(\alpha_j^{(i)})}{\Gamma(\alpha_j^{(i)})} + \log \tau_j \tag{4.31}$$

and

$$\frac{\partial g_i(\mathbf{X};\mathbf{\Lambda})}{\partial \beta_j^{(i)}} = \frac{\alpha_j^{(i)}}{\beta_j^{(i)}} - \tau_j \tag{4.32}$$

where $\Gamma'(\alpha_j^{(i)})$ denotes the derivative of $\Gamma(\alpha_j^{(i)})$ with respect to $\alpha_j^{(i)}$ and is computed with numerical differentiation. Adaptation of the duration modelling parameters is thus relatively easily integrated into the MCE framework.

4.4 Discriminative optimisation of linear model transformations

Linear transformation for speech model adaptation usually follows the maximum likelihood approach, leading to the well known MLLR algorithm or variants of it. In contrast to this, when linear transformation is applied to the speech pre-processing or feature extraction

stage, linear discriminant analysis (LDA) [110], principal component analysis (PCA) [38] or discriminatively optimised linear transformations using MCE [107] are commonly used. We do not explore feature space reduction or discriminative optimisation of the feature extraction process, since the techniques are liable to be database, or at least language specific. We, however, are interested in the application of discriminative methods in the optimisation of the linear transformation of the HMM model parameters between languages, as this may improve on the performance of maximum likelihood transformation estimators.

Rathinavelu [111] proposed applying the MCE/GPD method in optimising the parameters of a linear transformation of the trajectory parameters of a non-stationary state HMM. We independently arrived at the same method for the transformation of the Gaussian mean components of a mixture observation density HMM. If the transformation of the Gaussian mean components is given by

$$\hat{\boldsymbol{\mu}}_{jk} = \mathbf{W}\boldsymbol{\mu}_{jk}, \tag{4.33}$$

the observation probability of a state in the transformed HMM becomes (for diagonal covariance)

$$b_j^{(i)}(\mathbf{x}_t) = \sum_{k=1}^{M} c_{jk}^{(i)} \mathcal{N}[\mathbf{x}_t, \mathbf{W}\boldsymbol{\mu}_{jk}^{(i)}, \boldsymbol{\Sigma}_{jk}^{(i)}] = \sum_{k=1}^{M} \frac{c_{jk}^{(i)}}{(2\pi)^{(D/2)} \prod_{l=1}^{D} \sigma_{jkl}^{(i)}} e^{-\frac{1}{2}\sum_{l=1}^{D}(\frac{x_{tl} - \mathbf{w}_l \mu_{jk}^{(i)}}{\sigma_{jkl}^{(i)}})^2}. \tag{4.34}$$

Derivation of the MCE loss function with respect to the transformation matrix $\mathbf{W}$ then proceeds in a similar fashion to Equations 4.18 through 4.21. These equations give the partial derivative of the loss function with respect to any parameter of the mixture distribution in terms of the derivative of the log-observation probability with respect to that parameter. Therefore all that remains is to compute the derivative of the state log-observation probability density with respect to the transformation matrix. This is given for the lth row of

W by

$$\frac{\partial \log b_j^{(i)}(\mathbf{x}_t)}{\partial \mathbf{w}_l} = \frac{1}{b_j^{(i)}(\mathbf{x}_t)} \frac{\partial}{\partial \mathbf{w}_l} \Big[\sum_{k=1}^{K} c_{jk}^{(i)} \mathcal{N}[\mathbf{x}_t, \mathbf{W}\boldsymbol{\mu}_{jk}^{(i)}, \boldsymbol{\Sigma}_{jk}^{(i)}] \Big]$$
$$= \frac{1}{b_j^{(i)}(\mathbf{x}_t)} \Big[\sum_{k=1}^{K} c_{jk}^{(i)} \mathcal{N}[\mathbf{x}_t, \mathbf{W}\boldsymbol{\mu}_{jk}^{(i)}, \boldsymbol{\Sigma}_{jk}^{(i)}] \, (\frac{x_{tl} - \mathbf{w}_l \boldsymbol{\mu}_{jkl}^{(i)}}{{\sigma_{jkl}^{(i)}}^2}) \boldsymbol{\mu}_{jk}^{(i)} \Big]. \quad (4.35)$$

Examination of Equation 4.35 shows that there is a ${\sigma_{jkl}^{(i)}}^2$ term in the denominator, which may cause the update to be unstable even for a small update parameter due to the extremely large range of gradient values associated with a small variance component. A solution to this is to reduce the quadratic form to first order, or even to drop the variance term in the denominator of the gradient altogether. This heuristic solution can be better expressed in the GPD framework proposed by Juang & Katagiri [91] that caters for a positive definite matrix U_n as part of the update equation, which is then given by

$$\mathbf{w}_l(n+1) = \mathbf{w}_l(n) - U_n \epsilon_n \sum_{o=1}^{O} \sum_{c=1}^{M} 1(\mathbf{X}_o \in C_c) \frac{\partial l_c(\mathbf{X}_o; \boldsymbol{\Lambda})}{\partial \mathbf{w}_l} \Big|_{\boldsymbol{\Lambda}=\boldsymbol{\Lambda}_n}. \quad (4.36)$$

Choosing U_n to be a diagonal scaling matrix, with the average component variance for the lth dimension taking the lth position on the diagonal, provides a way of normalising the influence of the variance in the update equation. Note that this could also have been used in the MCE Gaussian mean update equation (Equation 4.14), but was not necessary since using the transformed mean value has exactly the same effect.

A choice has to be made with respect to the initial value $\mathbf{W}(0)$ of the transformation matrix. Using an identity matrix presents one option, but in light of the tendency of the gradient descent procedure to converge to local optima, a better choice is perhaps to use an MLLR estimate for the initial value. The optimisation process, however, is perhaps not as sensitive to the initial value, number of iterations and update parameter values because of the simple nature of the linear transformation process and because relatively fewer parameters are optimised. Experiments with the discriminatively optimised linear transform used MLLR initial estimates for the transformation and achieved improved performance

over standard MLLR transformation. However, since both MLLR and the discriminatively optimised linear transform did not produce very good performance in isolation, experiments in Chapters 6 and 7 only detail MLLR-based transformation results, mainly for comparison with MLLR-MAP results.

In the next section we discuss modifying the MCE loss and misclassification measures to associate varying cost or loss with different misclassification errors. Associating a cost with a particular misclassification indicates the importance (or lack of importance) of the misclassification and can improve the performance achievable with discriminative phoneme model optimisation by focusing on phoneme errors that have a high probability of leading to word errors.

4.5 Cost-based MCE

The Bayesian framework for classifier design [25] allows for the specification of a cost or risk ζ_{ij} associated with classifying a sample from class i as belonging to class j. In this sense, the standard implementation of MCE uses only a true-false cost function, considering in the misclassification measure (Equation 4.6) only the class to which an observation belongs as the true class and treating all other classes equally as false classes. When the true goal of a classifier is to achieve minimum phoneme misclassification, use of a zero-one cost function makes sense. In this case the MCE loss function closely approximates the empirical misclassification rate and presents a suitable function for optimisation. Generally, however, the goal of a classifier may be better expressed in terms of a more useful property such as word accuracy in continuous speech, or even at a more abstract level, in terms of how accurately the meaning of a speech utterance is expressed by a recognised phrase. In this section we consider ways of improving the MCE loss function to more accurately reflect the goal of the classifier. We start off with a previously discussed approach that implements discriminative training by comparing competing hypotheses.

4.5.1 String-level MCE

In order to improve the performance of discriminative training techniques for continuous speech applications, research was performed by Chou *et al.* [92] on a minimum string error rate implementation of MCE using N-best candidate strings. Even though the method targets a string-level loss function, optimisation occurs at the subunit (word or phoneme model) level, thereby indirectly also optimising classification performance of the subunits. The method has been shown to work well for closed vocabulary problems such as connected digit recognition. This is to be expected because it implements a task dependent word error rate based minimisation that compares possible in-vocabulary errors to the correct alignment and computes the update accordingly. String-level MCE is very useful when speech data is available that has been transcribed, but not labelled, since alignment information is not needed for the method. The method also may have an advantage over the standard approach in that recognition units are automatically aligned in sentence context with each other during training and can therefore take into account insertion and deletion errors in addition to substitution errors. However, McDermott [95] points out that string-level MCE effectively only considers regions of the speech input frame where there are differences in segmentation between the correct and competing hypotheses. Since only a limited number of hypotheses are typically decoded for computational reasons, only limited regions of each input frame are used to increase discrimination, whereas with phoneme-level MCE adaptation, many or all competing hypotheses (single HMMs) are considered for every phoneme segment, thereby better utilising the available data to increase class separation.

String-level MCE will not necessarily deliver optimal performance at the subunit level, as has been found [95] for continuous phoneme recognition. This can be explained by considering exactly what the effect of string-level training on the basic modelling units are. In LVCSR systems, phoneme models are usually used as the building blocks for composite speech units such as words. String-level MCE uses an N-best search to find the best competing hypotheses that differ in terms of word sequence from the correct hypothesis. Adaptation occurs only for models associated with these strings, thus predicating the adap-

tation of phoneme models on word confusability. The cost in terms of word error rate for classifying one phoneme as another is thus determined based on the whole word training data and used to adjust the boundary between the two phonemes. This is, however, a simplified view of how the method works. Typically, an exact pronunciation dictionary is not available, and therefore a training speech utterance may not exactly match the phoneme sequence of the correct word sequence. The acoustic models are thus adapted to also exhibit phonemic properties. This happens anyway if forced-alignment training is done, and can surely improve task specific word recognition performance, but care should be taken in predicting performance a task with a different vocabulary or grammar.

We have now discussed how string-level training can use a string and thus in effect a word error-based cost in adapting phoneme models. The method we discuss next shows a way of directly integrating a phoneme misclassification cost into the MCE framework, without having to perform an N-best search-based word level alignment.

4.5.2 Incorporating cost into the loss function

Reasons for applying modification of the phoneme models at the phoneme level rather than at the string or word level include:

- greater efficiency is achieved with (phoneme) model-level MCE versus string (word) level MCE,
- better vocabulary independence can be achieved because the cost matrix can be manipulated directly and is not dependent on the training speech utterances and
- the adaptation of each model can be controlled separately.

If the cost ζ_{ij} associated with the classification of a sample from class i as being of class j is known and is treated as a risk, the design of a minimum risk classifier in the Bayes sense can be attempted (determination of ζ_{ij} is the topic of a following section).

McDermott [86] suggested a modification of the MCE loss function to integrate a model-level cost function that ensures that the overall loss function reflects the empirical risk. For each training token the method weights the contribution from each false class by the risk associated with that false class. To incorporate the method into the MCE misclassification measure, the weight of each incorrect class j with respect to the total contribution of the incorrect classes is first expressed as

$$\omega_j(\mathbf{X};\mathbf{\Lambda},c) = \frac{e^{g_j(\mathbf{X};\mathbf{\Lambda})\eta}}{\sum_{f,f\neq c}^{M} e^{g_f(\mathbf{X};\mathbf{\Lambda})\eta}}, \tag{4.37}$$

where the correct class is c. The cost-based loss function can then be expressed in terms of the cost ζ_{cj} and the contribution $\omega_j(\mathbf{X};\mathbf{\Lambda},c)$ of each incorrect class, summed over all the incorrect classes by

$$l_c^*(\mathbf{X};\mathbf{\Lambda}) = \Big[\sum_{j,j\neq c}^{M} \zeta_{cj}\,\omega_j(\mathbf{X};\mathbf{\Lambda},c)\Big]\; l_c(\mathbf{X};\mathbf{\Lambda}). \tag{4.38}$$

Figure 4.1 shows graphically for a two class problem how the loss varies as a function of the position of an observed value, when different cost values are associated with the misclassification. The loss is a function of the relative correct versus incorrect class likelihoods and also the cost associated with misclassifying the correct class as the incorrect class. Since Equation 4.38 expresses the overall empirical risk, optimisation of the equation minimises the risk. If suitable estimates of the individual risks ζ_{cj} are available, the method may approximate minimum risk classification.

Unfortunately, no further details of the implementation of the method were published in [86]. Since we are interested in implementing the method we compute the derivative of the cost-based loss function $l_c^*(\mathbf{X};\mathbf{\Lambda})$ with respect to the class discriminant function $g_i(\mathbf{X};\mathbf{\Lambda})$,

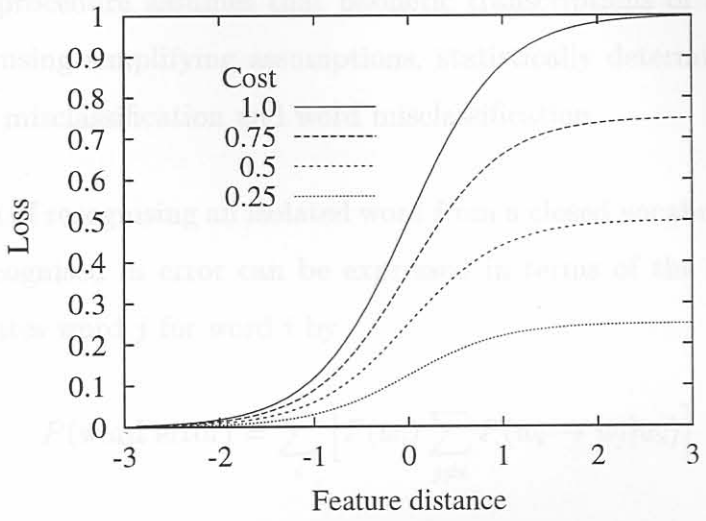

Figure 4.1: The loss l_c^* incurred as a function of the position of an observed value for various values of the misclassification cost when the Gaussian mean of the correct class is at -1 and the Gaussian mean of an incorrect class is at 1 and both Gaussians have unity variance

which is then given by

$$\frac{\partial l_c^*(\mathbf{X};\mathbf{\Lambda})}{\partial g_i(\mathbf{X};\mathbf{\Lambda})} = 1(i \neq c)\eta\omega_i(\mathbf{X};\mathbf{\Lambda},c)\left[\zeta_{ci} - \sum_{j,j\neq c}^{M} \zeta_{cj}\omega_j(\mathbf{X};\mathbf{\Lambda},c)\right] l_c(\mathbf{X};\mathbf{\Lambda}) + \left[\sum_{j,j\neq c}^{M} \zeta_{cj}\omega_j(\mathbf{X};\mathbf{\Lambda},c)\right]\frac{\partial l_c(\mathbf{X};\mathbf{\Lambda})}{\partial g_i(\mathbf{X};\mathbf{\Lambda})}, \quad (4.39)$$

where the partial derivative of the loss function with respect to the class discriminant function ($\frac{\partial l_c(\mathbf{X};\mathbf{\Lambda})}{\partial g_i(\mathbf{X};\mathbf{\Lambda})}$) is given by Equation 4.30. Since the derivation of the derivative of the discriminant function with respect to HMM, duration and transformation parameters have been given previously in this chapter, the process is now fully specified, except for the determination of the cost function itself, which is discussed next.

4.5.3 Estimating cost based on word error

We formulate a rather simple procedure for estimating the cost of misclassification for each phoneme pair that is based on the probability of the misclassification leading to a

word error. The procedure assumes that phonetic transcriptions of a large set of words are available and using simplifying assumptions, statistically determines the relationship between phoneme misclassification and word misclassification.

Given the problem of recognising an isolated word from a closed vocabulary, the probability that a word is recognised in error can be expressed in terms of the probability that the recogniser substitutes word j for word i by

$$P(\text{word error}) = \sum_i \left[P(w_i) \sum_{j \neq i} P(w_i \rightarrow w_j | w_i) \right] \tag{4.40}$$

where $P(w_i \rightarrow w_j | w_i)$ denotes the conditional probability that the substitution takes place and $P(w_i)$ is the *a priori* occurrence of word i. The next step is to condition the probability of a word substitution $w_i \rightarrow w_j$ on a specific phoneme substitution $\alpha_k \rightarrow \alpha_l$. The problem is greatly simplified by considering for each word pair only phoneme errors that change the first word along the optimal alignment path of the word pair to look more like the second word, hereafter termed *cross-word* phoneme errors. We define the phoneme misclassification cost ζ_{kl} to be the probability of a word error given that a specific substitution $\alpha_k \rightarrow \alpha_l$ of phoneme α_k by α_l occurs by:

$$\begin{aligned}
\zeta_{kl} &= P(\text{word error} | \alpha_k \rightarrow \alpha_l) \\
&= \frac{P(\text{word error}, \alpha_k \rightarrow \alpha_l)}{P(\alpha_k \rightarrow \alpha_l)} \\
&= \frac{\sum_i \left[P(w_i) \sum_{j \neq i} P(w_i \rightarrow w_j | w_i, \alpha_k \rightarrow \alpha_l)\, 1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\}) \right]}{\sum_i \left[P(w_i) \sum_{j \neq i} 1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\}) \right]} \\
&= \frac{\sum_i \left[P(w_i) \sum_{j \neq i} P(\text{word error} | \mathrm{d}(w_i, w_j), \#\text{substitutions} \geq 1)\, 1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\}) \right]}{\sum_i \left[P(w_i) \sum_{j \neq i} 1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\}) \right]}
\end{aligned} \tag{4.41}$$

where $1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\})$ is 1 when the $\alpha_k \rightarrow \alpha_l$ substitution match occurs in the optimal alignment of w_i and w_j and 0 otherwise, $\mathrm{d}(w_i, w_j)$ is the number of insertions, deletions and substitutions in the optimal alignment of w_i and w_j and the probability of a

word error, given the distance between the words and the fact that at least one substitution takes place is given by $P(\text{word error}|\text{d}(w_i|w_j), \#\text{substitutions} \geq 1)$.

The probability of a word error is thus defined to depend only on the phonetic distance $\text{d}(w_i, w_j)$ between the words, i.e. on the number of insertions, deletions and substitutions necessary to convert one word to the other. Independence of the cross-word phoneme errors is assumed and the number of cross-word phoneme errors then assumes a binomial distribution. When more than 50% of cross-word phoneme errors occur, a word substitution is assumed to take place and when exactly 50% of the cross-word phoneme errors occur, a 50% chance of a word substitution error is assumed. The probability of a word substitution is expressed by the summation of the binomial probabilities that half or more of the cross-word phoneme errors occur, taking into account that at least one phoneme error has occurred. The word substitution probability can then be expressed in terms of the inter-word phonetic distance n and the cross-word phoneme error probability p by the function

$$b(n,p) = \begin{cases} \sum_{m=\lceil n/2 \rceil}^{n} \binom{n-1}{m-1} p^{m-1}(1-p)^{n-m} & n \text{ odd} \\ 1/2\binom{n-1}{n/2-1} p^{n/2-1}(1-p)^{n/2} + \sum_{m=n/2+1}^{n} \binom{n-1}{m-1} p^{m-1}(1-p)^{n-m} & n \text{ even.} \end{cases} \tag{4.42}$$

Figure 4.2 shows graphically how the word substitution probability varies as a function of the inter-word phonetic distance for a number of cross-word phoneme error probabilities. Since cross-word phoneme errors only include phoneme errors that change the first word according to the optimal alignment path with the second word, the use of a relatively small value for the cross-word phone error probability is therefore applicable. We selected to use a value of 0.1 for the cross-word phoneme error probability.

Conditioning the probability of a word error only on phoneme substitutions (Equation 4.41) is perhaps too simplistic. We therefore extended the method to also consider the effect of phoneme insertions and deletions, by regarding them in the same way as substitutions. An insertion before or after α_k of α_l is considered a possible misclassification of speech

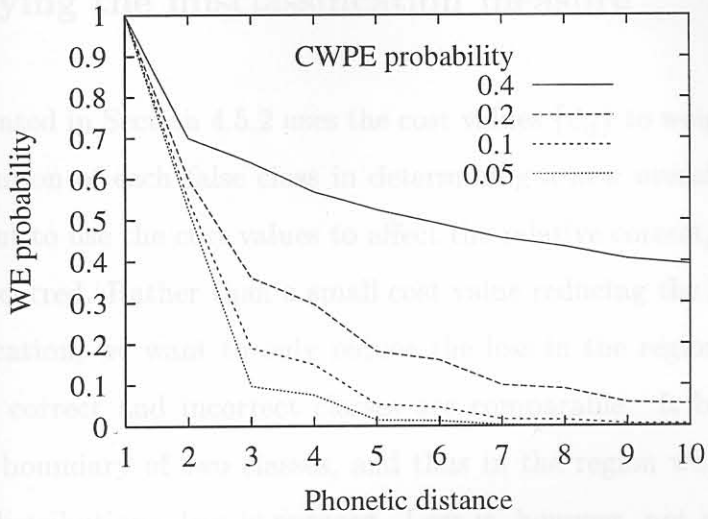

Figure 4.2: The word substitution error (WE) probability as a function of the inter-word phonetic distance for a number of values of the cross-word phoneme error (CWPE) probability, given that at least one substitution has occurred

data corresponding to α_k as α_l. Similarly, the deletion of α_k before or after α_l is also considered a possible misclassification of speech data corresponding to α_k as α_l. By associating a probability of $\frac{1}{2}$ with the mentioned insertions or deletions being caused by a $\alpha_k \rightarrow \alpha_l$ misclassification, the probabilities that these phoneme errors lead to word errors are incorporated into Equation 4.41 by the following equation:

$$\zeta_{kl} = P(\text{word error}|\alpha_k \rightarrow \alpha_l) + \frac{1}{2}P(\text{word error}|\alpha_k \uparrow \alpha_l) + \frac{1}{2}P(\text{word error}|\alpha_k \downarrow \alpha_l), \quad (4.43)$$

where $\alpha_k \uparrow \alpha_l$ means that α_l is inserted before or after α_k and $\alpha_k \downarrow \alpha_l$ means that α_k is deleted before or after α_l. Both $P(\text{word error}|\alpha_l \uparrow \alpha_k)$ and $P(\text{word error}|\alpha_k \downarrow \alpha_l)$ are computed in the same way as for $P(\text{word error}|\alpha_k \rightarrow \alpha_l)$ (Equation 4.41).

This concludes our discussion of the estimation of the word error-based phoneme misclassification cost. The resulting cost matrix can be used with the loss function method discussed in Section 4.5.2, or can be used with an alternative method that we discuss in the following section.

4.5.4 Modifying the misclassification measure

The method presented in Section 4.5.2 uses the cost values $\{\zeta_{ij}\}$ to weight for each training token the contribution of each false class in determining a new overall loss function. We may, however, want to use the cost values to affect the relative correct/incorrect likelihood at which loss is incurred. Rather than a small cost value reducing the total loss associated with a misclassification, we want to only reduce the loss in the region where the relative likelihoods of the correct and incorrect classes are comparable. It basically means that near the decision boundary of two classes, and thus in the region where overlap may occur between the distributions, loss is reduced. Loss is, however, not significantly reduced when the incorrect class likelihood is much higher than the correct class likelihood. This effectively shifts the loss function towards the incorrect class as the cost associated with a misclassification becomes lower.

Cost-based misclassification measure

In order to achieve the above, we present an approach to integrate the cost function (ζ_{kl}) into the MCE framework, based on a modification of the misclassification measure. The decision boundary of the misclassification measure is shifted with the value of the cost function. The new misclassification measure can then be expressed by

$$d_i^{\dagger}(\mathbf{X};\mathbf{\Lambda}) = -g_i(\mathbf{X};\mathbf{\Lambda}) + \log\left[\frac{1}{M-1}\sum_{j,j\neq i}^{M} e^{(\log\zeta_{ij}+g_j(\mathbf{X};\mathbf{\Lambda}))\eta}\right]^{1/\eta}. \tag{4.44}$$

The log cost $(\log\zeta_{ij})$ is added to the log-likelihood function, which is equivalent to multiplication of the likelihood function by the linear cost. Figure 4.3 shows graphically for a two class problem how the loss varies as a function of the position of an observed value, when different cost values are associated with the misclassification. It can be seen that the effect of the cost is to shift the loss function towards the incorrect class for lower cost.

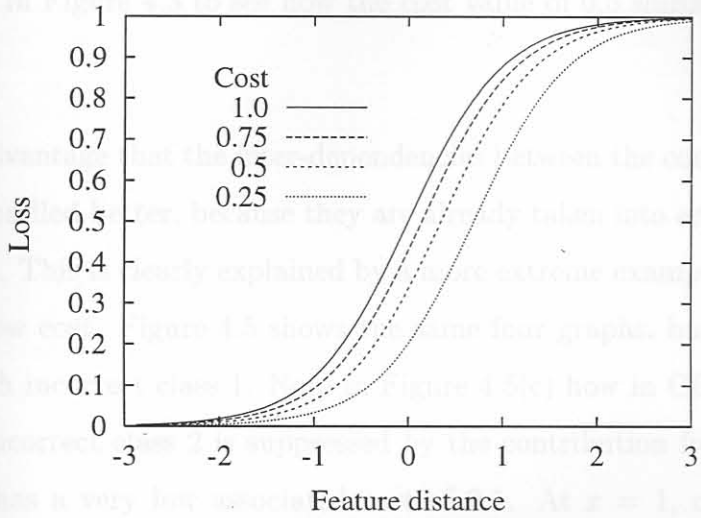

Figure 4.3: The loss $l_c^\dagger$ incurred as a function of the position of an observed value for various values of the misclassification cost used in the misclassification measure, when the Gaussian mean of the correct class is at -1 and the Gaussian mean of an incorrect class is at 1 and both Gaussians have unity variance

Comparison of modified misclassification measure with modified loss function approach

It is interesting to compare the working of the two approaches (Equations 4.44 and 4.38) to incorporating cost into the loss function. We refer to the approach of Section 4.5.2 that uses the cost to weight the loss function (4.38) as the cost-based loss function (CBLF) approach and to our approach (4.44) as the cost-based misclassification measure (CBMM) approach.

Figure 4.4 shows a three class problem with one correct class centred at -1 and two incorrect classes, centred at 1 and 3 and associated with misclassification costs of 0.5 and 1.0 respectively. Figure 4.4(a) shows the three Gaussian distributions, as well as the "average" incorrect class value as expressed by Equation 4.7 in the linear domain. Figure 4.4(b) also shows the three Gaussian distributions, but multiplied by their respective costs in the linear domain, as is effectively performed by CBMM in Equation 4.44 (by adding log-cost in the log domain). Figure 4.4(c) shows the shape of the loss function for the distributions in (a) achieved with CBLF (Equation 4.38). Figure 4.4(d) shows the loss CBMM associates with the class likelihood functions in (b). The contribution from incorrect class 1 can be

compared to that in Figure 4.3 to see how the cost value of 0.5 shifts the loss to the right in this case.

CBMM has the advantage that the inter-dependencies between the contribution of different false classes are handled better, because they are already taken into account in the misclassification measure. This is clearly explained by a more extreme example, in which incorrect class 1 has very low cost. Figure 4.5 shows the same four graphs, but with a cost of only 0.1 associated with incorrect class 1. Note in Figure 4.5(c) how in CBLF the contribution of the loss from incorrect class 2 is suppressed by the contribution from incorrect class 1, although class 1 has a very low associated cost of 0.1. At $x = 1$, one expects a loss in the region of 0.5 because the point is halfway between the true class and incorrect class 2, yet incorrect class 1 suppresses the loss. Figure 4.5(d) shows how in the CBMM approach the loss attributed to class 2 is only slightly suppressed by incorrect class 1 because the dependency between the incorrect classes in the misclassification measure is handled better.

Cost and reward-based misclassification measure

The goal of phoneme-level discriminative training should be the improvement of the overall system, of which the word error rate is a reasonably good measure. The method for estimating a word error-based phoneme misclassification cost and the integration of it into the discriminative training of phonemes provides a step in the right direction. String-level training, as we have discussed before, goes even a step further because it performs a degree of phonemic training - i.e. training based on what was supposed to have been said rather than for what was actually said. The method we propose next attempts to incorporate some phonemic information in the discriminative training procedure, while at the same time reducing the overall loss and thus the degree of adaptation that will occur.

The first step in the procedure is the extension of the word error-based phoneme misclassification cost estimation procedure to include a reward (negative cost) for a misclassification that may improve the word recognition rate. This is possible because different phonetic

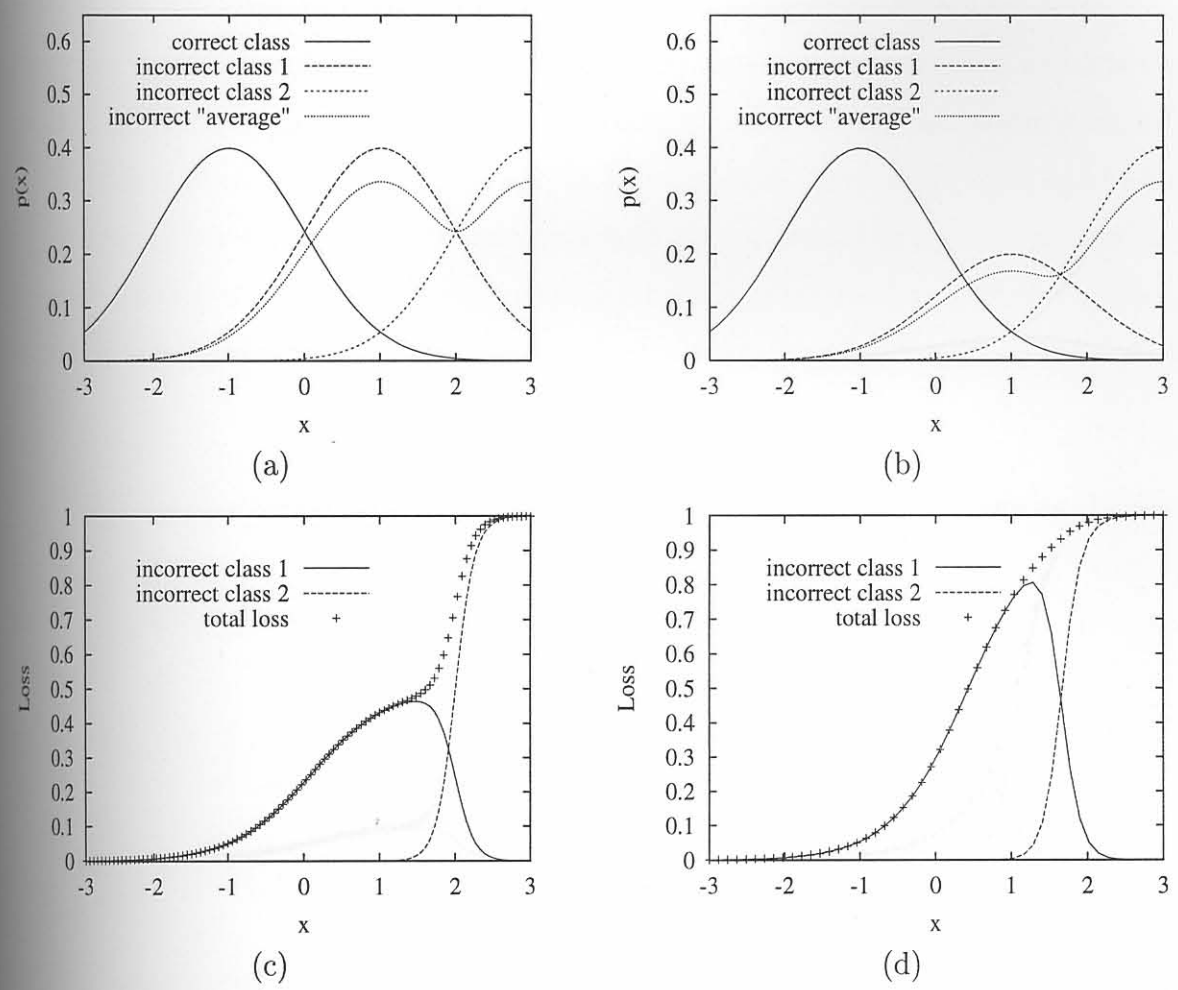

Figure 4.4: A comparison of two methods for computing the loss for a three class problem with a correct Gaussian mean at -1 and incorrect Gaussian means at 1 and 3 and with associated misclassification costs of 0.5 and 1 respectively, showing (a) the three distributions along with the "average" of the two incorrect classes, (b) the three distributions, modified according to the CBMM approach along with the "average" of the two incorrect classes, (c) the loss function according to the CBLF approach and (d) the loss function according to the CBMM approach

variants of the same word may occur in practice while the pronunciation dictionary for the word contains only a subset of the possibilities. The misclassification of one or more phonemes corresponding to the actual speech as the phonemes from the pronunciation dictionary may thus improve the overall word recognition rate. A procedure to estimate the expected reward associated with such misclassification can be derived by modifying

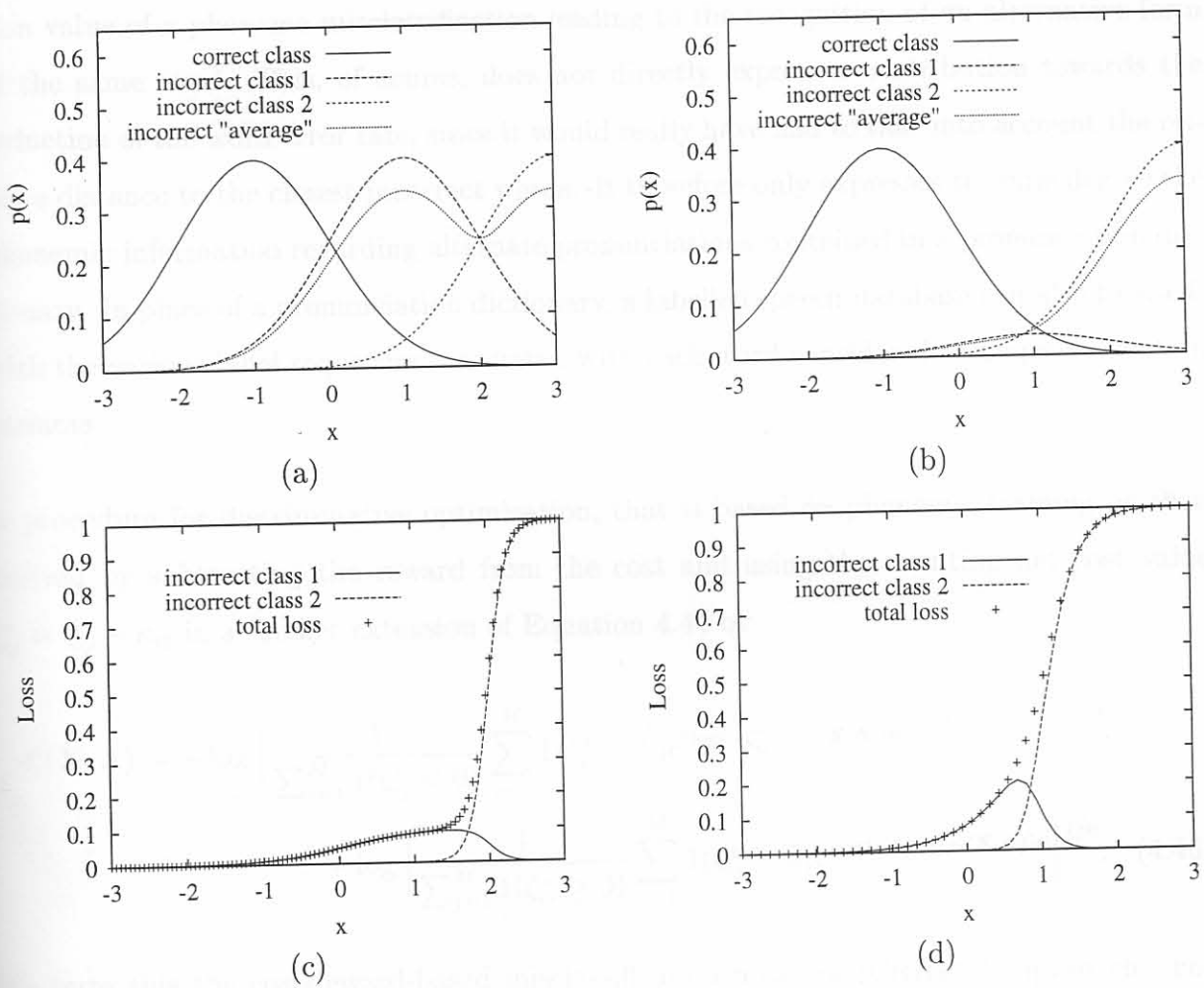

Figure 4.5: A comparison of two methods for computing the loss for a three class problem with a correct Gaussian mean at -1 and incorrect Gaussian means at 1 and 3 and with associated misclassification costs of 0.1 and 1 respectively, showing (a) the three distributions along with the "average" of the two incorrect classes, (b) the three distributions, modified according to the CBMM approach along with the "average" of the two incorrect classes, (c) the loss function according to the CBLF approach and (d) the loss function according to the CBMM approach

Equation 4.41 in the following way:

$$\kappa_{kl} = \frac{\sum_i \left[P(w_i) \sum_{j \neq i, w_j \in \Omega_i} P(\text{word subst.}|\mathrm{d}(w_i, w_j), \#\text{subst.} \geq 1)\, 1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\}) \right]}{\sum_i \left[P(w_i) \sum_{j \neq i} 1(\alpha_k \rightarrow \alpha_l \text{ in} \{w_i, w_j\}) \right]} \tag{4.45}$$

where Ω_i represents the phonetic variants of word i. Equation 4.45 estimates the expecta-

tion value of a phoneme misclassification leading to the recognition of an alternative form of the same word. This, of course, does not directly express a contribution towards the reduction of the word error rate, since it would really have had to take into account the relative distance to the closest incorrect words. It therefore only expresses to some degree the phonemic information regarding alternate pronunciations contained in a pronunciation dictionary. In place of a pronunciation dictionary, a labelled speech database can also be used, with the various label sequences associated with each word considered to be pronunciation variants.

A procedure for discriminative optimisation, that is based on phonemic training, is then derived by subtracting the reward from the cost and using the resulting net cost value $\zeta^*_{ij} = \zeta_{ij} - \kappa_{ij}$ in a further extension of Equation 4.44 by

$$d_i^{\dagger}(\mathbf{X};\mathbf{\Lambda}) = -\log\left[\frac{1}{\sum_{j=1}^{M} 1(\zeta^*_{ij}<0)}\sum_{j=1}^{M} 1(\zeta^*_{ij}<0)e^{(\log(-\zeta^*_{ij})+g_j(\mathbf{X};\mathbf{\Lambda}))\eta}\right]^{1/\eta} + \log\left[\frac{1}{\sum_{j=1}^{M} 1(\zeta^*_{ij}\geq 0)}\sum_{j=1}^{M} 1(\zeta^*_{ij}\geq 0)e^{(\log(-\zeta^*_{ij})+g_j(\mathbf{X};\mathbf{\Lambda}))\eta}\right]^{1/\eta}. \quad (4.46)$$

We term this the cost-reward-based misclassification measure (CRBMM) approach. For phoneme pairs with a net reward or negative net cost, i.e. $\zeta^*_{kl} < 0$, values of the parameters of model j will be adapted to increase the likelihood of observations from class k, thereby effecting phonemic training. It should, however, be noted that this approach reduces the empirical loss and therefore less adaptation will likely take place than for zero-one cost functions. The reason why this approach works may thus be rooted not only in the fact that it performs phonemic training, but in the fact that it reduces the loss associated with errors that have some positive or little negative effect, thereby stopping the MCE approach from changing ML estimated models to enforce rigid acoustic separation between phonetic classes.

The working of the method is shown in Figure 4.6 for the same three class problem with Gaussians centred at -1, 1 and 3, but with an associated net cost of -1, -0.1 and 1.0

respectively. Figure 4.6(a) shows that the net reward of 1.0 (net cost of -1.0) associated with correct class 1 effectively means that its likelihood function is unaffected, while the 0.1 net reward associated with correct class 2 means that its likelihood function is multiplied by 0.1 in the linear domain. The likelihood function of class 3 (the incorrect class) is unaffected by its cost of 1.0. The "average" likelihood of the classes with net reward (correct classes) is compared to the net loss class likelihood in the misclassification measure. Figure 4.6(b) shows the total loss incurred, as well as the portion of the total loss attributed (in the component derivatives) to class 1 and class 2. It can be seen that for large x, loss is attributed to class 2, which is then adapted rather than class 1, which has almost no contribution to the loss for large x.

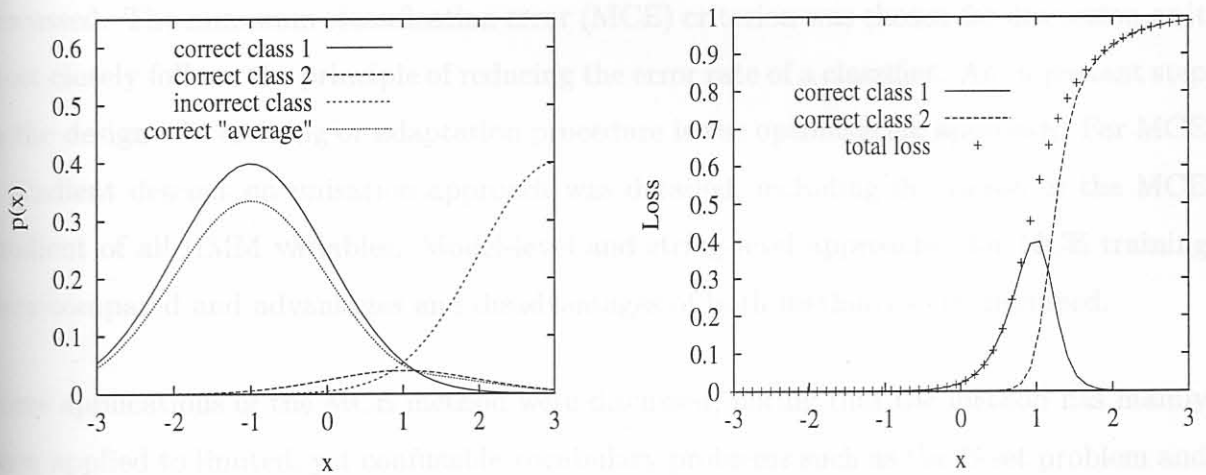

Figure 4.6: The CRBMM approach for computing the loss for a three class problem with Gaussian means at -1, 1 and 3 and with net rewards of 1.0 and 0.1 and a net loss of 1.0 respectively, indicating in (a) the three weighted distributions along with the average of the classes with net reward and in (b) the loss function as well as the contributions to the loss function by each of the classes with net reward

As far as the implementation of the procedure is concerned, the only derivation that changes is that of the misclassification measure. The derivative of the modified misclassification measure with respect to the class discriminant function becomes

$$\frac{\partial d_c^{\dagger}(\mathbf{X},\mathbf{\Lambda})}{\partial g_i(\mathbf{X};\mathbf{\Lambda})} = -\frac{1(\zeta_{ci}^* < 0)e^{[\log(-\zeta_{ci}^*)+g_i(\mathbf{X};\mathbf{\Lambda})]\eta}}{\sum_{j=1}^{M} 1(\zeta_{cj}^* < 0)e^{[\log(-\zeta_{cj}^*)+g_j(\mathbf{X};\mathbf{\Lambda})]\eta}} + \frac{1(\zeta_{ci}^* \geq 0)e^{[\log(-\zeta_{ci}^*)+g_i(\mathbf{X};\mathbf{\Lambda})]\eta}}{\sum_{j=1}^{M} 1(\zeta_{cj}^* \geq 0)e^{[\log(-\zeta_{cj}^*)+g_j(\mathbf{X};\mathbf{\Lambda})]\eta}}. \tag{4.47}$$

Note that this equation is also valid for the derivative of the misclassification measure of the CBMM approach (Equation 4.44) with respect to $g_i(\mathbf{X};\mathbf{\Lambda})$ by setting the correct class cost ζ_{ii} to -1.

4.6 Discussion

In this chapter we discussed the application of discriminative learning methods for the purpose of training and adapting parameters of speech recognition systems, continuous density HMMs in particular. The effect of the optimisation criterion on classifier design was discussed. The minimum classification error (MCE) criterion was chosen for discussion as it most closely follows the principle of reducing the error rate of a classifier. An important step in the design of a training or adaptation procedure is the optimisation approach. For MCE a gradient descent optimisation approach was detailed, including derivation of the MCE gradient of all HMM variables. Model-level and string-level approaches for MCE training were compared and advantages and disadvantages of both methods were discussed.

Some applications of the MCE method were discussed, noting that the method has mainly been applied to limited, yet confusable vocabulary problems such as the E-set problem and connected digit recognition. Only a limited number of medium to large vocabulary continuous speech recognition applications of MCE have been published. Adaptation performance of MCE was also discussed, with research indicating that better model initialisation, such as achieved by first performing MAP estimation, improves performance achieved with MCE and is better than ML adaptation in isolation.

Extensions to the standard MCE framework were presented, including discriminative adaptation of duration modelling variables, discriminative linear parameter transformation and word error-based phoneme adaptation approaches. The reason why the adaptation of all parameters, rather than say only Gaussian mean parameters are considered, is that cross-language adaptation may require significant adaptation, compared to perhaps the fine tun-

ing of models for a specific speaker. Alternative approaches for incorporating cost into the MCE framework were compared and an approach that also utilises reward in the misclassification measure was presented. The cost-based framework is of specific importance for cross-language adaptation since the phoneme inventory, context and acoustic separation between phonemes differ significantly between languages and adaptation should be able to address these issues efficiently for the target language.

In the next chapter we treat the issues involved in applying the techniques from speaker adaptation (Chapter 3) and discriminative learning (this chapter) for cross-language acoustic adaptation in detail.