DISCRIMINATIVE TRAINING TECHNIQUES IN HIDDEN MARKOV MODELS

Aravind Ganapathiraju

Department for Electrical and Computer Engineering
Mississippi State University, Mississippi State, MS 39762
ganapath@isip.msstate.edu

ABSTRACT

Maximum likelihood (ML) estimation has been primary optimization technique for hidden Markov model (HMM) parameter estimation. However, ML suffers for severe assumptions that includes the assumption that the underlying model is of a know form. ML estimation uses all the positive examples belonging to the class being modeled. This lack of discriminatory power has given rise to several estimation techniques that explicitly incorporate discriminative information into the optimization process. I n this paper we will present an overview of two commonly used discriminative training techniques for HMMs — Maximum Mutual Information and Minimum Classification Error. MMI incorporates discriminative information implicitly by using mutual information as the objective function. MCE does this explicitly. Implementation details and comparison of performance of the techniques on a small vocabulary task will also be presented.

1. INTRODUCTION

Maximum Likelihood (ML) is the most commonly used parameter estimation technique in hidden Markov models (HMM). The existence of efficient implementations in the form of the EM algorithm make the use of ML very attractive [10]. A drawback of the ML approach is that the model parameters are estimated based on the data belonging to that model only. It is independent of all the other models being estimated. This is however not the best way to improve recognition performance. Some form of discrimination needs to be added to the estimation process to improve the performance. Neural networks and support vector machines estimate parameters discriminatively [14, 15]. However they are not tractable to model temporal variation in their basic form. at the core of which is still an HMM parameter optimization problem.

Figure 1, shows a sample classification problem where using a ML approach is not the best for classification accuracy. The two classes are derived from two completely separable uniform distributions. ML is used to fit Gaussians to these classes and a simple Bayes classifier is built. However, we see that the decision threshold occurs inside the range of class 2. This means that the probability of error is significant. However if we were to simply recognize that the range of data points in class 1 is less than 3.3 and that no data point in class 2 occurs within this range, we can achieve perfect classification. This makes for a strong case to pursue explicit discrimination techniques for HMM parameter estimation.

In this paper we will review the theory and implementation of Maximum Mutual Information (MMI) and Minimum Classification Error (MCE) estimation techniques [1,2,9]. Performance on a small vocabulary task, TIMIT, will also be discussed [13].

2. MAXIMUM LIKELIHOOD

The goal of the HMM parameter estimation process is to maximize the likelihood of the data given the model, traditionally known as Maximum Likelihood (ML) estimation [5]. In effect ML tries to maximize the a posteriori probability of the training data given the model. Note that this implies that other models are not part of this optimization process. One of the most compelling reasons for the success of ML and HMMs has been the existence of iterative methods to estimate the parameters while guaranteeing convergence. Expectation-Maximization (EM) is one algorithm

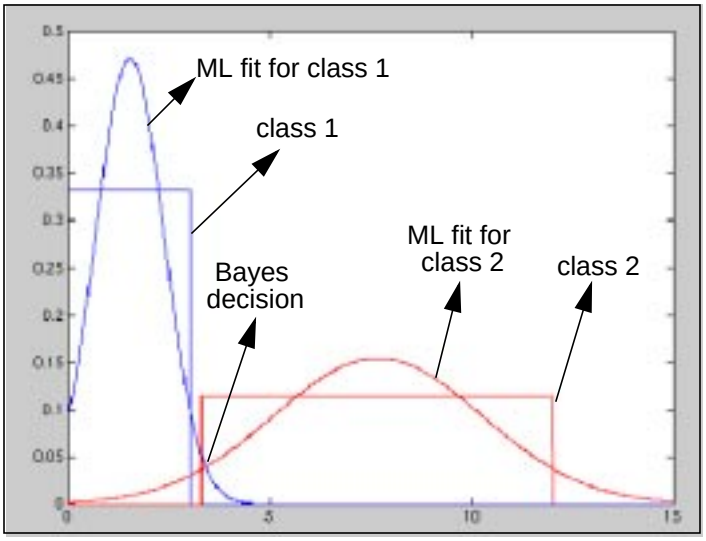

Figure 1. An example where maximum likelihood estimation fails to perfectly classify completely separable data. Notice the Bayes decision threshold is inside the distribution on the right. A discriminator that learns the information that class 1 does not go beyond the value 3.3 can do a perfect job of classification.

that is used extensively to perform ML estimation.

2.1. EM Theorem and Maximum Likelihood

If,

$$\sum_t P_{\theta'}(t|y)\log P_{\theta}(t|y) > \sum_t P_{\theta'}(t|y)\log P_{\theta'}(t|y) \qquad (1)$$

then,

$$P_{\theta}(y) > P_{\theta'}(y) \qquad (2)$$

The gist of the above theorem is that, if we start with a model θ' and find a model θ such that equation 1 is satisfied, then the observed data y is more probable under the model θ than under θ' [7]. This is a very powerful theorem in that it guarantees convergence at least to a local minimum. In the above formulation t is the intermediate random variable that depends on the model parameter settings. For example, t could be the state sequence in an HMM which is not something we observed. The terms on the LHS and RHS of equation 1 can be represented as the auxiliary functions $Q(\theta', \theta)$ and $Q(\theta', \theta')$. Since we are maximizing the auxiliary function in the EM framework, the parameter update equations can be obtained by differentiating $Q(\theta', \theta)$ with respect to each of the parameters and setting the derivative to zero. When t, is chosen as the state sequence, the EM formulation is called the Baum-Welch algorithm [5]. A detailed explanation of the update equations for each of the HMM parameters can be found in [1].

The ML approach works well if the assumption that the form of the parametric probability model that computes $P(O|M)$ is the same as the true underlying distribution. This is a very restrictive assumption is many cases and assumes the availability of a large amount of training data to estimate the parameters of a complex process like speech.

2.2. Optimization

In order to compute the probability of the state sequence in the HMMs en route to computing the likelihood of the data given the model, the forward and backward probabilities are used and form the sufficient statistics for this estimation process [5]. To talk about these computations we need to start with the definition of the HMM.

HMMs are finite state machines in their basic form. They differ from regular finite state machines in that each state also has a probability of emitting a symbol. Apart from this there is a probability distribution representing the probability of a transition from one state to another [12]. The complete description of the model can be provided using the following quantities:

- N — the number of states
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(\boldsymbol{o})\}$, where $\boldsymbol{o}$ is the input observation vector

The output probability distribution gives the probability of observing a vector in the given state. The most commonly used form of the output distribution is a multivariate Gaussian. Other distributions like Laplacians have been used in some systems [16]. The multivariate Gaussian can be written as:

$$b_j(\boldsymbol{o}_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} \exp\left(-\frac{1}{2}(\boldsymbol{o}_t - \mu_j)' \Sigma_j^{-1} (\boldsymbol{o}_t - \mu_j)\right) \quad (3)$$

where $\boldsymbol{o}_t$ is the observation vector at time t and the subscript j indicates that the Gaussian under consideration belongs to the jth state. The following formulation assumes that the input consists of T observation vectors.

2.2.1. Forward Probability

The forward probability gives us the probability of generating the observations from time 1 to t and the model ending in state j at time t.

$$\alpha_j(t) = Pr(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t, x(t) = j) \quad (4)$$

The above computation can be efficiently done using the following recursive formulation.

$$\alpha_j(1) = a_{ij} b_j(\boldsymbol{o}_1) \quad (5)$$

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij}\right] b_j(\boldsymbol{o}_t), \quad \text{for} \quad 1 < t \le T$$

and, (6)

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T) a_{iN} \quad (7)$$

2.2.2. Backward Probability

The backward probability is the probability of generating the observations from time $t+1$ to T if the model was in state j at time t.

$$\beta_j(t) = Pr(\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T | x(t) = j) \quad (8)$$

Similar to the forward probability computation, a recursive formulation exists for the backward probability computation.

$$\beta_i(t) = a_{iN} \quad (9)$$

$$\beta_i(t) = \sum_{i=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1), \quad \text{for} \quad 1 \le t < T$$

and, (10)

$$\beta_1(1) = \sum_{i=2}^{N-1} a_{1j} b_j(\boldsymbol{o}_1) \beta_j(1) \quad (11)$$

2.2.3. Utterance Likelihood

The total utterance probability, P, given the model M, can be written in terms of α and β as:

$$P(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_j(t) \beta_j(t) \text{ or,} \quad (12)$$

2.3. Parameter Update

The forward and backward probabilities defined above form the sufficient statistics for the estimation process. The EM algorithm tells us that the likelihood of the data given the model is maximized if the expected value of the probability of the state sequence is maximized. This conditional probability is computed in terms of the forward and backward probabilities. The gradients are computed and parameters updated base on the gradient values. [7] contains a detailed derivation for the update equations for all the parameters in an HMM.

3. MAXIMUM MUTUAL INFORMATION

MMI tries to incorporate the discriminative information into the optimization process implicitly. As will be seen in the following sections, its theory and implementation parallel ML significantly. The main difference is the parameter update process which is based on a gradient descent approach unlike ML estimation.

3.1. Motivation

The mutual information, I, between variables X and Y is defined as the average amount of uncertainty about the knowledge of X given knowledge of Y [11]. Mathematically this can be defined as:

$$I(X;Y) = H(X) - H(X/Y) \quad (13)$$

The conditional entropy of X given Y is given by

$$\sum_{x,y} P(x,y)\log P(x/y) = -E[\log P(x/y)] \quad (14)$$

Having defined mutual information, we now pose the speech recognition problem in the same framework. Let W, O denote the random variables corresponding to the words and observation vectors. The uncertainty in the word sequence given the acoustic observations is the conditional entropy of W given O,

$$H(W/O) = H(W) - I(W;O). \quad (15)$$

Note that we do not know $P(w,o)$ in general and need to estimate a parametric fit. The conditional entropy of the words given the acoustic observations can be shown to confirm to the following inequality:

$$H_\lambda(W/O) \geq H(W/O), \quad (16)$$

where λ denotes a particular parametric fit to the actual distribution [1, 2, 11]. The equality holds only if $P_\lambda(w/o) = P(w/o)$, we can get an optimal estimate of the conditional distribution. This minimization can also be done as a maximization of the mutual information and hence the name maximum mutual information (MMI) to this optimization technique [1, 2].

The objective function L_{MMI} for the MMI estimation of the parameters is nothing but the mutual information of the words given the acoustic observations under the parametric distribution λ.

$$L_{MMI}(\lambda) = I_\lambda(X;Y) = H_\lambda(W) - E[\log P_\lambda(w/o)] \quad (17)$$

Replacing the expectations by the sample averages and assuming the training data to comprise of R utterances,

$$L_{MMI}(\lambda) = -\frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(w_r) + \frac{1}{R}\sum_{r=1}^{R}\log\frac{P_\lambda(o_r|M_r)P_\lambda(w_r)}{P_\lambda(o_r)} = \frac{1}{R}\sum_{r=1}^{R}\{\log P_\lambda(o_r|M_r) - \log P_\lambda(o_r)\} \quad (18)$$

In the above equation w_r is the word sequence in the r^{th} utterance with a corresponding composite model M_r. o_r are the set of observation vectors corresponding to the r^{th} utterance. The first term in the above equation is equivalent to the ML optimization criterion and the second term is what makes this a discriminative framework.

3.2. Optimizing Mutual Information

Equation 18 defines the objective function to be optimized to achieve maximum mutual information. The first term in the equation is the standard ML computation that can be done efficiently via the forward-backward probabilities for the correct transcription of each of the training utterances.

The second term is where the discrimination information is added. It can be rewritten as,

$$P_\lambda(a_r) = \sum_{\hat{w}} P_\lambda(a_r/M_{\hat{w}})P_\lambda(\hat{w}) \quad (19)$$

where $\hat{w}$ is the set of all word sequences. For small application this set of all word sequences can be compactly represented using a grammar or a graph. The forward-backward probabilities are computed using this grammar instead of just the correct transcription. These probabilities can now be used for optimizing the objective function.

In the case of large vocabulary applications, representing the word sequences with a grammar can be an ominous task, if not impossible. In such cases a sub-optimal solution is chosen where an N-best list or a word-graph is used instead.

3.3. Gradient Descent and Iterative Solutions

The formulation in 18 is that is that of unconstrained optimization. Typically this is approached using iterative procedures [1, 9]. Several procedures have been developed over the years. The following sections discuss the commonly use iterative algorithms to find the maximum using a gradient approach. All the methods compute the gradient of the objective function with respect to the parameters of the HMM, namely the mean, variance and transition probabilities.

Let $\boldsymbol{O}$ be the observation sequence and let M be the models under consideration with a parameter set λ. The likelihood of the observation sequence given the model is defined as,

$$L_\lambda(O/M) = \log P_\lambda(O/M) \quad (20)$$

such that,

$$\frac{\partial L_\lambda}{\partial \lambda} = \frac{1}{P_\lambda(\boldsymbol{O}|M)} \frac{\partial}{\partial \lambda} P_\lambda(O/M) \tag{21}$$

The posterior probability can be written in terms of the α and β as,

$$P_\lambda = P_\lambda(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_j(t)\beta_j(t). \tag{22}$$

In order to introduce the other parameters in the model into the above equation, we can write it as,

$$P_\lambda = \sum_{t=1}^{T} \sum_{j=1}^{N} \left\{ \sum_{i=1}^{N} \alpha_i(t) a_{ij} \right\} b_j(\boldsymbol{o}_t)\beta_j(t) \tag{23}$$

Transition Probability

The transition probabilities need to be handled carefully in order to guarantee that the transitions out of any state sum to unity. For this reason, a regularization function is used to redefine the transitions as,

$$a_{ij} = \frac{f_a(h_{ij})}{\sum_k f_a(h_{ik})} \text{ and } f_a(x) = e^x. \tag{24}$$

which is also known as softmax is some literature [15]. Then,

$$\frac{\partial a_{ij}}{\partial h_{ik}} = a_{ij}(\delta_{kj} - a_{ik}), \tag{25}$$

where δ is the Kroneker delta.

Since P depends on a_{ij}'s, applying the chain rule gives,

$$\frac{\partial P_\lambda}{\partial h_{ik}} = \sum_j \frac{\partial P_\lambda}{\partial h_{ij}} \left(\frac{\partial a_{ij}}{\partial h_{ik}} \right). \tag{26}$$

Differentiating 23 with respect to a_{ij} and using it along with 25 in 26 yields,

$$\frac{\partial P_\lambda}{\partial h_{ik}} = \sum_t \sum_j \alpha_i(t-1) a_{ij}(\delta_{kj} - a_{ik}) b_j(\boldsymbol{o}_t)\beta_j(t) \tag{27}$$

Mean

In order to get the contribution of the means towards the likelihood, we need to start with the definition in equation 3. Differentiating this equation with respect to the d^{th} component of the mean of distribution corresponding to the j^{th} state, we get,

$$\frac{\partial}{\partial \mu_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t) \left\{ \frac{o_{td} - \mu_{jd}}{\sigma^2_{jd}} \right\} \tag{28}$$

From 3, we know that,

$$\frac{\partial P_\lambda}{\partial b_j(\boldsymbol{o}_t)} = \sum_{t=1}^{T} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} \beta_j(t) \tag{29}$$

Using the chain rule for partial derivatives, we get,

$$\frac{\partial P_\lambda}{\partial \mu_{jd}} = \sum_{t=1}^{T} C(t, j) b_j(\boldsymbol{o}_t) \left\{ \frac{o_{td} - \mu_{jd}}{\sigma^2_{jd}} \right\}, \text{ where,} \tag{30}$$

$$C(t, j) = \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} \beta_j(t). \tag{31}$$

Variance

As mentioned earlier, for simplicity we assume diagonal covariances in these derivations. In the case of diagonal covariances, we need to constrain the values such that all of them are positive. In order to convert the constrained set to an unconstrained set (as we did with the transitions), we use the following regularization.

$$\sigma_{jd}^2 = f(z_{jd}) \text{ and } f(x) = e^x. \tag{32}$$

We can start first by looking at the gradient of the output distribution with respect to the variance.

$$\frac{\partial}{\partial \sigma^2_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t) \frac{1}{2} \left\{ \frac{(o_{td} - \mu_{jd})^2}{(\sigma^2_{jd})^2} - \frac{1}{\sigma^2_{jd}} \right\} \tag{33}$$

Using 32, we can convert 33 in terms of the regularization variable z as,

$$\frac{\partial}{\partial z_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t) \frac{1}{2} \left\{ \frac{(o_{td} - \mu_{jd})^2}{\sigma^2_{jd}} - 1 \right\} \tag{34}$$

Using 29 and the chain rule for partial derivatives, we get,

$$\frac{\partial P_\lambda}{\partial z_{jd}} = \sum_{t=1}^{T} C(t,j) b_j(\boldsymbol{o}_t) \left\{ \frac{(o_{td} - \mu_{jd})^2}{(\sigma^2_{jd})^2} - 1 \right\} \quad (35)$$

These quantities can be easily extended to include multiple instances of the model and multiple mixture Gaussians.

3.3.1. Steepest Descent

The steepest descent method is the simplest and most commonly used iterative technique to minimize a multivariate function. At the end of each iteration, the values of the parameters are updated in the direction in which the objective function decreases the most. The change in the parameter value is a constant proportion of the gradient of the objective function with respect to the parameter. The constant proportion is commonly referred to as the *learning rate*.

This procedure can be concisely written as

$$x_{k+1} = x_k - \eta \nabla L_{MMI}(x_k), \quad (36)$$

where, x_k is the value of the parameter x in the k^{th} iteration, $L_{MMI}(x_k)$ is the gradient of the objective function with respect to the parameter and η is the learning rate. The value of the learning rate is typically computed as a line search over $L_{MMI}(x_k - \eta \nabla L_{MMI}(x_k))$. This can be an expensive process if the number of parameters being estimated is large.

3.3.2. Momentum

Steepest descent can have slow convergence depending on the optimizing surface. The direction of descent and the magnitude of the gradient play a role in defining the convergence properties of the optimization process. At places where the function surface is fairly flat, the gradient vector has a small magnitude and reaching the optimum may be slow. Similarly when the function is steep, the gradient may be large and the parameter can be updated to a value that may result in oscillation. Momentum attempts to solve the above problems by introducing a new term in the update equation in steepest descent.

The new update equation a momentum term is

$$\Delta x_k = (1-\zeta)\eta \nabla L_{MMI}(x_k) + \zeta \Delta x_{k-1}. \quad (37)$$

We see that this formulation uses the amount of change in the parameter value during the previous iteration on top of the typical steepest descent update. So, in the case where consecutive iterations have the same sign for the gradient, the update amount is larger and vice versa.

3.3.3. Local Optimization

Local optimization methods are a specific form of the descent methods where individual learning rates are applied to each parameter. Another feature of these iterative methods is the ability to change the learning rates over time. Some of the heuristics used to change the learning rates include:

- when the sign of the gradient with respect to a parameter changes sign for several iterations, the learning rate is decreased
- when the sign of the gradient with respect to a parameter does nor change for several frames, the learning rate is increased

3.3.4. RProp

RProp differs from the procedures discussed above in that only the sign of the gradient is used to update the parameter values instead of also using the magnitude. In RProp the learning rates are varied with iterations following a simple update procedure.

$$\eta_k = \begin{cases} min(1.2\eta_{k-1}, \eta_{max}) & \text{if same direction} \\ max(0.5\eta_{k-1}, \eta_{min}) & \text{if opp. direction} \\ \eta_{k-1} & \text{otherwise} \end{cases} \quad (38)$$

The change in the parameter value is then given by

$$\Delta x_k = sign[\nabla L_{MMI}(x_k)]\eta_k \quad (39)$$

3.4. Implementation

Figure 2 shows procedure involved in estimating models in the MMI framework. Notice that the ML models are also obtained as a by-product at the end of this procedure. In large vocabulary applications, instead of the recognition grammar, N-best lists or word-graphs are used where the number of alternate word sequences is limited. Using the N-best lists is actually good since only the most confusable data is used to improve discrimination.

4. MINIMUM CLASSIFICATION ERROR

Thus far we have seen parameter estimation using the tradition ML technique and the discriminative MMI technique. However, notice that neither of the two explicitly tried to optimized the primary goal of a speech recognizer, classification rate. This section looks at a parameter estimation technique geared to handle this.

For a given loss function $l_k(\boldsymbol{x}, \Lambda)$ where $\boldsymbol{x}$ is the feature vector belonging to the class C_k and Λ is the parameter set, the aim of parameter estimation using the Minimum Classification Error (MCE) principle is to optimize the overall expected loss $L(\Lambda)$,

$$L(\Lambda) = \sum_{k=1}^{M} P(C_k) \int l_k(\boldsymbol{x}, \Lambda) p(\boldsymbol{x}/C_k) d\boldsymbol{x} \tag{40}$$

However we do not have access to the probability distributions used above. Amari's generalized probabilistic theorem comes to our rescue here [17].

4.1. Generalized Probabilistic Descent

For an infinite sequence of random samples $\boldsymbol{x}_t$ and step size sequence c_t that satisfies the conditions

$$1. \sum_{t=1}^{\infty} c_t \rightarrow \infty, \text{ and} \tag{41}$$

$$2. \sum_{t=1}^{\infty} c_t^2 < \infty \tag{42}$$

adapting the system parameters according to

$$\Lambda_{t+1} = \Lambda_t - c_t U \nabla l_k(\boldsymbol{x}_t, \Lambda_t) \tag{43}$$

converges with a probability of one to a local

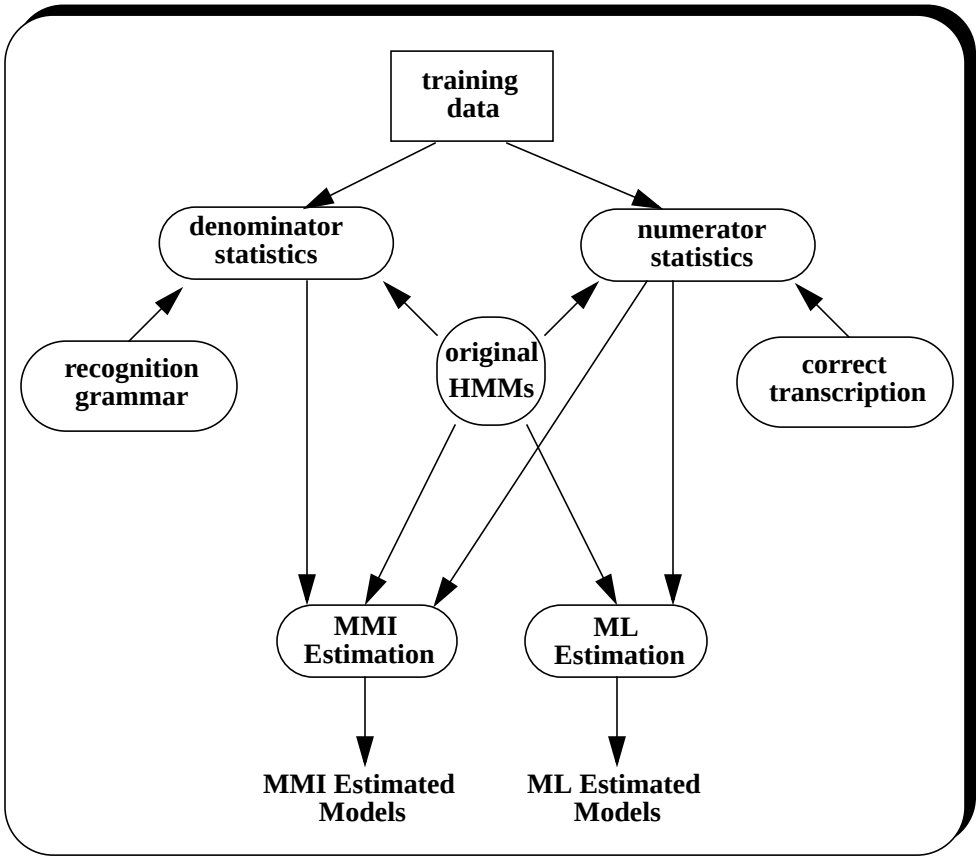

Figure 2. Implementation of the MMI estimation process. Note that only the left-half of the figure involves the new computations corresponding to MMI estimation. The right-half is the standard ML estimation process. Note the need for a recognition grammar for MMI estimation process.

minimum of $L(\Lambda)$. The key here is that the overall loss is never computed but we optimize based on the local loss $l_k(\boldsymbol{x}, \Lambda)$.

4.2. Loss Functions

As seen above, the goal of MCE can be achieved via optimization involving the local loss functions. In general there are certain desirable properties for loss functions:

- since the problem is that of minimizing classification errors, near-binary loss functions are desirable
- continuous functions
- loss functions need to be first order differentiable to apply GPD

The commonly used loss function with all the above features is the sigmoid function:

$$l(d) = \frac{1}{1 + e^{-\alpha d}} \tag{44}$$

where d is the misclassification error measure.

The misclassification error measure can be defined in terms of the discriminant functions of the k classes, C_k we would like to discriminate better as a result of the MCE optimization process. In typical speech recognition applications these classes are either words in the vocabulary or the phonemes in the system. Typically misclassifications for the k^{th} class will add 1 to d_k and a correct classification will subtract 1.

$$d_k(\boldsymbol{x}, \Lambda) = -g_k(\boldsymbol{x}, \Lambda) + \left[\frac{1}{L-1}\sum_{j \neq k} g_j(\boldsymbol{x}, \Lambda)^{-\psi}\right]^{-\frac{1}{\psi}} \tag{45}$$

where we assume the that this an L-class problem and ψ controls the contribution of each misclassification towards the error metric. Note that when ψ is large, the most confusable class contributes the most to the summation. This fits in well with the N-best list processing paradigm or the Viterbi approximation commonly used in many speech recognition systems . Note that in the current discussion we have not yet related the parameters of the HMM to the objective function defined in equation 40. The discriminant function for the k^{th} class, g_k, relates them together.

4.3. Implementation

There can be several definitions for the discriminant function. The only requirement is that the provides a distance metric to compare classes. Traditionally likelihood has been used for this purpose for HMMs. The computation of the likelihood is done as the probability of all possible state sequences Θ^p for the given data. In a closed form, for one particular state sequence this can be written as

$$f(\boldsymbol{x}_1^T, \theta^p \big| \Lambda) = \prod_{t=1}^{T} a_{\theta_{t-1}^p \theta_t^p} \cdot b_{\theta_t^p}(\boldsymbol{x}_t) \tag{46}$$

where a and b are the model parameters. Using the above definition of the likelihood, the discriminant function for the j^{th} class can be defined as

$$g_j(\boldsymbol{x}_1^T, \Lambda) = \log\left[\sum_p [f(\boldsymbol{x}_1^T, \theta^p \big| \Lambda)]^{\xi}\right]^{\frac{1}{\xi}} \tag{47}$$

Note that when ξ is large, the most probable state sequence dominates the summation and approaches the Viterbi framework.

We see how the quantities defined so far finally end up relating the estimated loss, L our objective function, to the model parameters a and b. Using the chain rule for partial derivatives, we can find the gradient of the loss with respect to the model parameters. At this point the optimization process is similar to what we do in MMI estimation.

To look at the flow of the training procedure, let us look at an application where MCE is applied at the word level, i.e. the classes are the words in the vocabulary. We make an important assumption here that the discriminant function uses normalized

# of mixtures	MLE	MMI	MCE
1	44.0	39.0	39.0
4	37.6	34.8	33.8
8	35.2	32.4	32.7
24	33.3	30.2	31.3

Table 1. comparison of ML, MMI and MCE on phone classification in TIMIT

probabilities in that we can compare scores for instances of a word k with different durations.

We start with N-best lists. State-level alignments for the N-best lists are obtained. The N-best lists at the timing information is used to obtain the word instances that the word k is confused with. These instance are used to compute $d_k(\boldsymbol{x}, \Lambda)$. This is used to compute the loss $l_k(\boldsymbol{x}, \Lambda)$. GPD theorem is then used to update the parameters for the models that went into defining the word k.

5. PERFORMANCE

Though it is obvious that ML is not the best optimization criterion, the discriminative techniques incur a heavy computational overhead for the gains in performance. Often, to reduce the computational overhead, sub-optimal solutions like using the Viterbi alignments or N-best list processing.

Most gains from using the above discussed estimation techniques have been reported on tasks ranging from small to medium vocabularies. We present here the performance on TIMIT as reported in [1] and [9]. Table 1 compares the three reestimation paradigms on the TIMIT data [13]. MMI and MCE models consistently do better models estimated using ML. However the gains reduce as the number of mixture components are increased. the reason for MMI models doing better than MCE is that MCE models has only their means and variances updated unlike MMI models where mixture weights and transitions were also updated.

6. SUMMARY

We have seen how discriminative information can be made a part of the optimization process to estimate the model parameters in HMMs. Theory and implementation issues involved in applying MMI and MCE paradigms have been presented in detail. Though the discriminative techniques discussed here are attractive theoretically, they suffer from high computational expense compared to the traditional ML estimation process. The discriminative techniques have been to perform better than ML on a TIMIT phone classification experiment. In order to apply these techniques to large vocabulary applications several sub-optimalities have to be pursued that include the use of Viterbi state alignments for likelihood computation and N-best list processing as a substitute for recognition grammars.

7. REFERENCES

[1] V. Valtchev, *Discriminative Methods in HMM-based Speech Recognition*, Ph. D. Thesis, University of Cambridge, UK, 1995.

[2] Y. Normandin, *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem*, Ph. D. Thesis, McGill University, Canada, 1991.

[3] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043-3054, 1992.

[4] P.C. Woodland and D.R. Cole, "Optimizing Hidden Markov Models using Discriminative Output Distributions," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April 1991.

[5] L. E. Baum et. al., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Finite State Markov Model Chains," *Annals of Mathematical Statistics*, vol. 41, pp. 164-171, 1970.

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

[7] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Massachusetts, USA, 1997.

[8] J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing, New York, USA, 1993.

[9] E. McDermott, *Discriminative Training for Speech Recognition*, Ph. D. Thesis, Waseda University, Japan, 1997.

[10] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum Likelihood Estimation from Incomplete Data," *Journal of the Royal Statis-*

tical Society, vol. 39, no. 1, pp. 1-38, 1977.

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

[12] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[13] W.M. Fisher, et. al., "The DARPA Speech Recognition Research Database: Specifications and Status," *Proceedings of the DARPA Speech Recognition Workshop*, 1986.

[14] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, NY, USA,1995.

[15] H.A. Bourlard and N. Morgan, *Connectionist Speech Recognition — A Hybrid Approach*, Kluwer Academic Publishers, Boston, USA, 1994.

[16] J.L. Gauvain, et. al., "The LIMSI 1995 Hub3 System," *Proceedings of the DARPA Speech Recognition Workshop*, pp. 105-111, Harriman, NY, USA, February 1996.

[17] S. Amari, "A Theory of Adaptive Pattern Classifiers," IEEE Transactions on Electronic Computers, vol. EC-16, no. 3, pp. 299-307, 1967.