Matrix Updates for Perceptron Training of Continuous Density Hidden Markov Models

Chih-Chieh Cheng CHC028@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego

Fei Sha FEISHA@USC.EDU
Department of Computer Science, University of Southern California

Lawrence K. Saul SAUL@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego

Abstract

In this paper, we investigate a simple, mistake-driven learning algorithm for discriminative training of continuous density hidden Markov models (CD-HMMs). Most CD-HMMs for automatic speech recognition use multivariate Gaussian emission densities (or mixtures thereof) parameterized in terms of their means and covariance matrices. For discriminative training of CD-HMMs, we reparameterize these Gaussian distributions in terms of positive semidefinite matrices that jointly encode their mean and covariance statistics. We show how to explore the resulting parameter space in CD-HMMs with perceptron-style updates that minimize the distance between Viterbi decodings and target transcriptions. We experiment with several forms of updates, systematically comparing the effects of different matrix factorizations, initializations, and averaging schemes on phone accuracies and convergence rates. We present experimental results for context-independent CD-HMMs trained in this way on the TIMIT speech corpus. Our results show that certain types of perceptron training yield consistently significant and rapid reductions in phone error rates.

1. Introduction

Continuous density hidden Markov models (CD-HMMs) are widely used in modern systems for automatic speech recognition (ASR) (Huang et al., 2001). The parameters of these models must be estimated from phonetically transcribed speech. The simplest approach to this problem is maximum likelihood estimation (MLE), which attempts to maximize the joint likelihood of the training data. At best, however, CD-HMMs provide only an approximate model of the tremendous variability observed in real speech. Moreover, the parameters in CD-HMMs must be estimated from limited amounts of training data. In this operating regime, MLE does not generally yield model parameters that minimize the error rate for ASR. This realization has led researchers to explore other methods for parameter estimation.

A great deal of research in ASR has focused on discriminative training of HMMs (Bahl et al., 1986; Nádas, 1983; Juang & Katagiri, 1992). Perhaps the most popular framework for discriminative training is maximum mutual information estimation (MMIE). MMIE attempts to compute model parameters that maximize the conditional probability of the correct phonetic transcription given the observed speech. More recently, building on the successes of support vector machines in the machine learning community, a number of researchers in ASR have also explored large margin methods for discriminative training of CD-HMMs (Jiang et al., 2006; Li et al., 2007; Yu et al., 2007; Sha & Saul, 2009).

In ASR, a major challenge of discriminative training arises from the combinatorially large number of possible phonetic transcriptions per speech utterance. To succeed, discriminative methods must separate the likelihood of the correct decoding from all incorrect hypotheses. The need to consider incorrect hypotheses makes discriminative training much more computationally intensive than MLE. MMIE has been successfully applied to large vocabulary ASR by using lattices to provide a compact representation of likely incorrect hypotheses (Woodland & Povey, 2000). Never-

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

theless, the scaling of discriminative methods to larger and larger speech corpora remains an important problem for ongoing research.

A similar problem of scaling confronts researchers in machine learning, whose algorithms must deal with data sets of ever increasing size. The demands of large-scale applications have led to a resurgence of interest in *online* learning algorithms (Bottou & LeCun, 2004; Bottou & Bousquet, 2008). These algorithms update model parameters after the presentation of each labeled example, thus eliminating the need to store or manipulate the entire data set in memory. Not only are these online algorithms simpler to implement and more feasible for large-scale learning, but in many cases they converge more quickly and attain better (generalization) performance than their batch counterparts.

One of the simplest and oldest online learning algorithms is the perceptron (Rosenblatt, 1958). An exciting line of recent work has generalized the perceptron algorithm to discriminative training of discrete HMMs (Collins, 2002). The perceptron algorithm for discrete HMMs combines simple additive updates with Viterbi decoding of training examples. On problems of part-of-speech tagging and base noun phrase chunking, this algorithm outperformed other leading discriminative approaches.

Motivated by the potential of this approach for ASR, in this paper we investigate a similarly inspired online learning algorithm for CD-HMMs. CD-HMMs for ASR are parameterized in terms of the means and covariance matrices of multivariate Gaussian emission densities. The online updating of these parameters raises several issues that do not arise in perceptron training of discrete HMMs. For instance, perceptron updates in CD-HMMs can violate the positive definiteness of covariance matrices, thus requiring further computation to maintain these constraints. Our main contribution (in sections 2.4–2.5) is to propose a particular reparameterization of CD-HMMs that lends itself very well to perceptron training. We present experimental results for CD-HMMs trained in this way on the TIMIT speech corpus, a standard data set for phoneme recognition benchmarks. We systematically compare the effects of different matrix parameterizations, initializations, and averaging schemes on recognition accuracies and convergence rates. Our results reveal the particular formulation of perceptron training that yields the most consistently significant and rapid reductions in recognition error rates.

The remainder of the paper is organized as follows. Section 2 describes our formulation of perceptron training in CD-HMMs. Section 3 presents experimental results that reveal the effects of different parameterizations, matrix factorizations, initializations, and averaging schemes. Finally, section 4 summarizes our most important findings and discusses various directions for future work.

2. Model

In this section, we begin by reviewing CD-HMMs and introducing our notation. We then present the perceptron algorithm for CD-HMMs in its most general terms. Finally, we consider how to reparameterize CD-HMMs in a more sensible way for perceptron training and highlight various issues that arise from this reparameterization.

2.1. Background

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

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

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

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

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

$$\mathbf{s}^* = \text{argmax}_s \log \mathcal{P}(\mathbf{s}, \mathbf{x}; \Theta). \tag{3}$$

The inference in eq. (3) depends on the parameters of the CD-HMM, which we collectively denote by Θ. The right hand side of eq. (3) can be computed efficiently by dynamic programming. In particular, of all possible sequences of hidden states, the Viterbi algorithm recursively constructs the one with the highest log-likelihood.

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

$$\Theta_{\text{ML}} = \text{argmax}_\Theta \sum_{n=1}^{N} \log \mathcal{P}(\mathbf{y}_n, \mathbf{x}_n; \Theta). \tag{4}$$

ML estimates of the parameters in CD-HMMs may be computed by the Expectation-Maximization (EM) algorithm. The EM algorithm monotonically increases the log-likelihood in eq. (4) with each update, scales well to large data sets, and does not involve any tuning parameters. All these properties make it very attractive as a starting point for ASR. However, one particular drawback of ML estimation is that maximizing the joint likelihood in eq. (4) does not generally minimize the error rate of the recognizer. To achieve this goal, we therefore seek a mistake-driven method for parameter estimation in CD-HMMs.

2.2. Perceptron Training

Discriminative training of CD-HMMs has a long history in ASR (Bahl et al., 1986; Nádas, 1983; Juang & Katagiri, 1992), and new work continues to appear in this area. The fundamental idea behind discriminative training is to seek parameters that explicitly minimize the error rate rather than attempting to model the data itself. Discriminative training of CD-HMMs is more complicated than ML estimation for several reasons: (i) log-likelihoods must be computed not only for desired state sequences, but also for competing ones; (ii) most update rules involve some form of gradient descent, requiring careful tuning of learning rates; (iii) convergence is not generally as fast as the EM algorithm for ML estimation.

Mindful of these issues, we have explored an online method for discriminative training of CD-HMMs based on the perceptron algorithm (Rosenblatt, 1958) and its recent use in discrete HMMs (Collins, 2002). We begin by describing our framework at a high level, then give further details (in sections 2.3–2.5) relating to the issue of parameterization.

We start by defining a so-called discriminant function over observation and hidden state sequences:

$$\mathcal{D}(\mathbf{x},\mathbf{s}) = \log\mathcal{P}(s_1) + \sum_{t>1}\mathcal{P}(s_t|s_{t-1}) + \sum_t \log\mathcal{P}(x_t|s_t). \tag{5}$$

The discriminant function is essentially the logarithm of the joint probability distribution in CD-HMMs, eq. (1). However, for discriminative training, we may adapt the parameters of the CD-HMM in such a way that they no longer define a properly normalized joint distribution. In particular, we need not enforce sum-to-one constraints on the rows of the transition matrix nor the mixture weights of GMMs.

In terms of the discriminant function, eq. (5), the target state sequence y_n will be correctly inferred if $\mathcal{D}(\mathbf{x}_n,\mathbf{y}_n)$ exceeds $\mathcal{D}(\mathbf{x}_n,\mathbf{s})$ for all competing state sequences $\mathbf{s}$. To estimate parameters that have this property, we consider the following online update rule:

$$\Theta \leftarrow \Theta + \eta\frac{\partial}{\partial\Theta}\left[\mathcal{D}(\mathbf{x}_n,\mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n,\mathbf{s}^*_n)\right], \tag{6}$$

where $\mathbf{s}^*_n$ denotes the most likely state sequence returned by the Viterbi decoding in eq. (3) and $\eta > 0$ is a carefully chosen learning rate. Eq. (6) is a mistake-driven update rule that only adjusts the parameters when the Viterbi decoding returns an incorrect answer $\mathbf{s}^*_n \neq \mathbf{y}_n$. The gradient in eq. (6) computes the fastest search direction in parameter space to close the gap in the discriminant function between $\mathcal{D}(\mathbf{x}_n,\mathbf{y}_n)$ and $\mathcal{D}(\mathbf{x}_n,\mathbf{s}^*_n)$. As in perceptron training, we update the parameters of the CD-HMM in an iterative fashion, looping through all the training examples until either the algorithm converges or no longer reduces the average number of classification errors.

In general, perceptron learning will not converge to a fixed set of parameter estimates if the training examples cannot be perfectly classified. However, convergence to a fixed set of parameter estimates can be obtained by averaging the results of perceptron training from different updates of the training examples (Freund & Schapire, 1999). In practice, this sort of averaging also appears to yield better results (Gentile, 2002) by damping fluctuations in the decision boundary that occur during training. Let $\Theta^{(j)}$ represent the parameter estimates after the perceptron update in eq. (6) has been applied for the j^{th} time. We compute the averaged parameters $\tilde{\Theta}^{(r)}$ after r updates as:

$$\tilde{\Theta}^{(r)} = \frac{1}{r}\sum_{j=1}^{r}\Theta^{(j)}. \tag{7}$$

Note that these averaged estimates are not themselves used during training; they are only computed after training, then used for the classification of new test examples.

2.3. Parameterization of GMMs

Conventionally, CD-HMMs are parameterized in terms of their state transition probabilities and the means and covariance matrices of their GMMs. The choice of parameterization plays an important role in perceptron training. For example, consider the update rules for the mixture weights $\mathcal{P}(c|s)$ and the diagonal elements of the covariance matrices Σ_{sc}. Simple additive updates to parameters may not preserve their nonnegativity, which is necessary for the discriminant function in eq. (5) to be well-defined for all possible observation and state sequences. More generally, the choice of parameterization can significantly affect the rate of convergence of perceptron training, as well as the nature of the averaging in eq. (7).

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

The GMMs in CD-HMMs are conventionally parameterized in terms of the mixture weights $\mathcal{P}(c|s)$, means μ_{sc}, and covariance matrices Σ_{sc} associated with different hidden states and mixture components. In fact, the most straightforward perceptron updates are given in terms of the log-mixture weights $\nu_{sc} = \log \mathcal{P}(c|s)$ and inverse covariance matrices Σ_{sc}^{-1}. In particular, the mixture weights are best updated in the log domain to ensure that they remain nonnegative. Moreover, it is simpler to compute the derivatives in the perceptron update rule, eq. (6), with respect to the inverse covariance matrices Σ_{sc}^{-1} than the covariance matrices Σ_{sc}. For the GMM parameters in CD-HMM, these considerations lead to perceptron updates of the form:

$$\begin{aligned}
\nu_{sc} &\leftarrow \nu_{sc} + \eta \frac{\partial}{\partial \nu_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right], && (8)\\
\mu_{sc} &\leftarrow \mu_{sc} + \eta \frac{\partial}{\partial \mu_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right], && (9)\\
\Sigma_{sc}^{-1} &\leftarrow \Sigma_{sc}^{-1} + \eta \frac{\partial}{\partial \Sigma_{sc}^{-1}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right]. && (10)
\end{aligned}$$

The last update in eq. (10) can violate the constraint that the inverse covariance matrix Σ_{sc}^{-1} must be positive definite; when this happens, the zero or negative eigenvalues of the updated matrix must be thresholded to some small positive value so that individual Gaussian distributions remain normalizable (or at least bounded). Though simple in concept, the updates in eqs. (8–10) do not work as simply as they appear. Note that the GMM parameters in these updates have different scales and dimensionalities, suggesting that gradient-based discriminative training might require careful tuning of multiple different learning rates in order to succeed. Alternatively, a common strategy is to only optimize the mean parameters of GMMs.

2.4. Reparameterization of GMMs

In this paper, following an earlier approach used in large margin CD-HMMs (Sha & Saul, 2009), we investigate a reparameterization that aggregates the mixture weight, mean, and covariance matrix parameters associated with each Gaussian mixture component into a single augmented matrix. Let $\gamma_{sc} = -\log[\mathcal{P}(c|s)/(2\pi)^d|\Sigma_{sc}|]$ denote the log of the scalar prefactor that weights each Gaussian mixture component. Then for each Gaussian mixture component, consider the matrix:

$$\Phi_{sc} = \begin{bmatrix} \Sigma_{sc}^{-1} & -\Sigma_{sc}^{-1}\mu_{sc} \\ -\mu_{sc}^\top \Sigma_{sc}^{-1} & \mu_{sc}^\top \Sigma_{sc}^{-1}\mu_{sc} + \gamma_{sc} \end{bmatrix}. \qquad (11)$$

In eq. (11), the upper left block of the matrix Φ_{sc} is simply the inverse covariance matrix Σ_{sc}^{-1}, while the other elements of Φ_{sc} are determined by the interaction of the mean μ_{sc} and covariance matrix Σ_{sc}. Note that in terms of this matrix, we can rewrite eq. (2) as:

$$\mathcal{P}(x|s) = \sum_c e^{-\frac{1}{2}z^\top \Phi_{sc} z} \quad \text{where} \quad z = \begin{bmatrix} x \\ 1 \end{bmatrix}. \qquad (12)$$

We can use the reparameterization in eqs. (11–12) to adapt the matrices Φ_{sc} by perceptron training, as opposed to the GMM parameters in the previous section. In this way, we can replace the three separate updates in eqs. (8–10) by the single update:

$$\Phi_{sc} \leftarrow \Phi_{sc} + \eta \frac{\partial}{\partial \Phi_{sc}} \left[\mathcal{D}(X_n, y_n) - \mathcal{D}(X_n, s_n^*)\right]. \qquad (13)$$

Like the earlier update in eq. (10) for the inverse covariance matrix, the update in eq. (13) can violate the constraint that the matrix Φ_{sc} must be positive semidefinite. When this happens, the updated matrix must be projected back onto the cone of positive semidefinite matrices.

As written, the update in eq. (13) effectively removes the constraint that the Gaussian distributions are properly normalized or even of bounded variance. Note that for a properly normalized Gaussian distribution, the bottom diagonal matrix element of Φ_{sc} in eq. (11) is completely determined by the mean μ_{sc} and covariance matrix Σ_{sc}. However, in discriminative training, we can update these matrix elements independently, no longer enforcing normalization constraints on each Gaussian mixture component. Unlike the earlier update in eq. (10) for the inverse covariance matrix, we can also allow the matrix Φ_{sc} to have strictly zero eigenvalues. In particular, though eq. (2) is not defined for singular covariance matrices, eq. (12) is perfectly well defined for all positive semidefinite matrices $\Phi_{sc} \succeq 0$. Thus the perceptron update in eq. (13) can learn to use unnormalized Gaussians with unbounded variance if they do indeed lead to fewer classification errors.

2.5. Matrix factorizations

The update in eq. (13) has the potentially serious drawback that it can violate the constraint that the matrices Φ_{sc} are positive semidefinite. Unlike the constraints of normalizability or bounded variance discussed in the last section, these constraints are important to enforce: otherwise a particular state s and mixture component c could be deemed more and more likely even as observed acoustic feature vectors deviated further and further away from its estimated mean μ_{sc}. Though updated matrices can be projected back into the cone of positive semidefinite matrices whenever these constraints are violated, projected gradient methods tend to exhibit considerably slower convergence than unconstrained methods.

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

$$\Phi_{sc} = \Lambda_{sc}\Lambda_{sc}^{T} \tag{14}$$

makes explicit that Φ_{sc} is positive semidefinite. With this factorization, we can replace the update in eq. (13) by

$$\Lambda_{sc} \leftarrow \Lambda_{sc} + \eta \frac{\partial}{\partial \Lambda_{sc}} \left[\mathcal{D}(X_n, y_n) - \mathcal{D}(X_n, s_n^{*})\right], \tag{15}$$

in which the square matrices Λ_{sc} (of the same size as Φ_{sc}) are completely unconstrained.

The update in eq. (15) has potential advantages and disadvantages. As a form of unconstrained optimization, it has the potential advantage of faster convergence since it does not involve a projected gradient step. On the other hand, it has the potential disadvantage of creating an optimization landscape with more local minima. In particular, note that for the special case in which each Gaussian mixture model has only one mixture component, the difference of discriminant functions is actually linear in the matrices Φ_{sc}. This simple optimization landscape is lost with the factorization in eq. (15). Our experiments in section 3.2 attempt to determine which potential advantages and disadvantages of this matrix factorization are realized in practice.

We note that the factorization in eq. (14) is not unique. While a matrix square root satisfying eq. (14) can be computed by singular value decomposition, the matrix Λ_{sc} is not uniquely determined unless additional constraints are imposed. One way to obtain a unique factorization is by constraining Λ_{sc} to be positive semi-definite; however, such a constraint is precisely what we hoped to finesse by factorizing the matrix Φ_{sc} in the first place. Another way to obtain a unique factorization – the Cholesky factorization – is by constraining Λ_{sc} to be a lower triangular matrix. In section 3.2, we evaluate and present results for two ways of updating the matrices Λ_{sc}: one that constrains them to be lower triangular, and one that does not.

The factorization in eq. (14) raises another issue related to the averaging of parameter estimates as in eq. (7). For training, we can update the matrices Φ_{sc} directly by eq. (13) or indirectly by eq. (15). However, the best approach for training does not necessarily correspond to the best approach for testing with smoothed parameter estimates. Using the notation of eq. (7), one approach is to average the parameter estimates for Φ_{sc} as:

$$\tilde{\Phi}_{sc}^{(r)} = \frac{1}{r}\sum_{j=1}^{r} \Phi_{sc}^{(j)} = \frac{1}{r}\sum_{j=1}^{r} \Lambda_{sc}^{(j)}\Lambda_{sc}^{(j)\top}. \tag{16}$$

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

$$\tilde{\Phi}_{sc}^{(r)} = \tilde{\Lambda}_{sc}^{(r)}\tilde{\Lambda}_{sc}^{(r)\top}, \text{ where } \tilde{\Lambda}_{sc}^{(r)} = \frac{1}{r}\sum_{j=1}^{r} \Lambda_{sc}. \tag{17}$$

In section 3.4, we evaluate and present results for both of these types of averaging.

3. Experiments

We performed experiments on the TIMIT speech corpus (Lamel et al., 1986). The speech signals in this corpus have been manually segmented and aligned with their phonetic transcription. We adopted the same front end as recent benchmarks for phoneme recognition on this data set (Sha & Saul, 2009). As is common for ASR, we computed 39-dimensional acoustic feature vectors of mel-frequency cepstral coefficients on sliding windows of speech. Finally, we followed the standard partition of the TIMIT corpus, yielding roughly 1.1 million, 120K, and 57K frames respectively for training, test, and holdout data.

We experimented with all the methods described in section 2 for perceptron training of CD-HMMs. The CD-HMMs were trained to minimize the number of phonetically misclassified frames in each utterance. The CD-HMMs had 48 hidden states (one per phone) and GMMs that varied in size from one to eight mixture components. In calculating the errors, we follow the standard of mapping 48 phonetic classes down to 39 broader categories (Lee & Hon, 1988).

We evaluated the performance of each CD-HMM by comparing the hidden state sequences inferred by Viterbi decoding to the "ground-truth" phonetic transcriptions provided by the TIMIT corpus. We report two types of errors: the frame error rate (FER), computed simply as the percentage of misclassified frames, and the phone error rate (PER), computed from the edit distances between ground truth and Viterbi decodings (Lee & Hon, 1988). While the perceptron update in eq. (6) is designed to minimize the frame error rate (which is approximated by differences between discriminant functions aggregated over frames), the phone error rate provides a more relevant metric for ASR.

Perceptron training of CD-HMMs raises several issues that do not arise in perceptron training of discrete HMMs. Our experiments addressed three main issues: (i) how should the GMMs be parameterized, in the same way as for MLE (section 2.3), or by aggregating the parameters for each mixture component into a single matrix (section 2.4)? (ii) how should we enforce the positive semidefiniteness constraints on matrix parameters, by projected gradient methods in the original parameter space or by reparameterizing the matrices using singular value decompositions or Cholesky factorizations (section 2.5)? (iii) in which parameter space should we average to obtain smoothed parameter estimates for testing (section 2.5)? Our experimental results provide fairly definitive answers to these questions.

Table 1. Frame and phone error rates for CD-HMMs of varying size, as obtained by maximum likelihood (ML) estimation, perceptron training (PT), and popular batch methods for discriminative training. The batch results for conditional maximum likelihood (CML) and minimum classification error (MCE) are reproduced from previous benchmarks (Sha & Saul, 2009). The left column shows the number of mixture components per GMM.

#	Frame Error Rate (%)		Phone Error Rate (%)			
mix	ML	PT	ML	PT	CML	MCE
1	39.3	30.0	42.0	35.2	36.4	35.6
2	37.1	27.6	38.6	33.2	34.6	34.5
4	31.4	26.0	34.8	31.2	32.8	32.4
8	28.1	26.5	32.5	31.9	31.5	30.9

3.1. Overall benefits of perceptron training

We begin by reporting results that confirm the well-known benefits of discriminative training and online learning. Table 1 compares the best performing CD-HMMs obtained by maximum likelihood (ML) estimation to the best performing CD-HMMs obtained by perceptron training (PT). The latter used the matrix update in eqs. (14–15) and the averaging scheme in eq. (16). The results show that perceptron training leads to significant reduction in both frame and phone error rates for all model sizes. The improvements in frame error rates are larger than the improvements in phone error rates; this discrepancy reflects the fact that the perceptron update more closely tracks the Hamming distance (not the edit distance) between target and Viterbi phone sequences. For reference, Table 1 also shows previously published benchmarks (Sha & Saul, 2009) from the two most popular batch approaches to discriminative training. It is interesting that for all but the largest model size, online perceptron training outperforms these batch approaches.

3.2. Benefits of reparameterization

As noted in section 2.3, the conventional parameters of GMMs have different scales and dimensionalities, making it difficult to apply the perceptron updates in eqs. (8-10). For example, in the simple case where each GMM has one mixture component, the discriminant function in eq. (5) has a quadratic dependence on the mean parameters but a linear dependence on the inverse convariance matrices. In practice, it is often necessary (and difficult) to tune separate learning rates for such different types of parameters. Our experiments bore out these difficulties. The left column of Table 2 shows our best results from discriminative training with this parameterization. In fact, these results were obtained by updating just the mixture weights and mean vectors of the CD-HMMs; despite extensive experimentation, we were unable to obtain further improvements by updating the inverse covariance matrices in parallel or even while holding the other parameters fixed. Our results are consistent with previous anecdotal observations in ASR: in particular, the common "folklore" is that most of the performance gains in discriminative training are due to optimizing the mean parameters in GMMs.

Table 2. Frame error rates from perceptron training of CD-HMMs with different parameterizations: updating (ν, μ, Σ^{-1}) in eqs. (8-10) versus updating Φ in eqs. (11–13).

#	Frame Error Rate (%)	
mix	(ν, μ, Σ^{-1})	Φ
1	35.7	32.2
2	37.0	31.5
4	33.6	30.7
8	32.9	30.4

Table 3. Frame error rates of perceptron training from the update in eq. (13) versus the update in eq. (15). For the latter, we studied two different forms of matrix factorization, one using singular value decomposition (SVD), one using Cholesky factorization. For each result, the number of sweeps through the training data is shown in parentheses.

#	Frame Error Rate (%)		
mix	Φ	Λ-SVD	Λ-Cholesky
1	32.2 (243)	30.0 (7)	33.1 (142)
2	31.5 (258)	27.6 (4)	31.9 (135)
4	30.7 (296)	26.0 (11)	32.5 (7)
8	30.4 (131)	26.5 (28)	31.4 (18)

The reparameterization in section 2.4 greatly simplifies both the form of the discriminant function, eq. (12), and the resulting updates for perceptron training. The results in the rightmost column of Table 2 reveal the benefits of this approach. These results were obtained using the reparameterization in eq. (11), the update in eq. (13), and the averaging in eq. (7). Note that perceptron training in the parameters Φ_{sc} from eq. (11) leads to significantly lower frame error rates across all model sizes.

3.3. Benefits of matrix factorization

We also experimented with the matrix factorization in eq. (14). Updating the matrices Λ_{sc} by eq. (15) led to the results shown in Table 3. The middle column shows the results when the matrices Λ_{sc} were unconstrained and initialized by singular value decomposition; the right column shows the results when the matrices Λ_{sc} were constrained to be lower diagonal and initialized by Cholesky factorization. For comparison, the left column repeats the results from Table 2 for updating the matrices Φ_{sc} by eq. (13). Note how the unconstrained factorization in eq. (14) leads to consistent further improvements beyond those obtained by the reparameterization in eq. (11). The factorization also leads to much faster convergence as measured by the num-

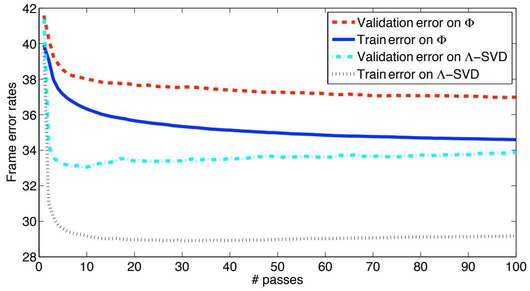

Figure 1. Comparison of perceptron training with and without the matrix factorization in eq. (14). See text for details.

bers of sweeps through the training data (shown in parentheses). Finally, as an additional benefit, the factorized update also avoids the extra computation required to project the updated parameters Φ_{sc} back into the space of positive semidefinite matrices.

Fig. 1 graphically illustrates the much faster convergence of perceptron training using the matrix factorization in eq. (14). The figure compares the frame error rates on the training and validation sets during training for the top left (Φ) and middle (Λ-SVD) results in Table 3. When updating the matrices Λ_{sc} using eq. (15), the training error drops rapidly, and the CD-HMM appears to start overfitting after just a few sweeps through the training data. By contrast, when updating the matrices Φ_{sc} using eq. (13), the training and holdout error rates drop much more slowly.

3.4. Benefits of averaging

Parameter averaging is an effective technique for reducing the fluctuations inherent to online learning. Table 4 demonstrates the benefits of parameter averaging in the setting of CD-HMMs, where it often leads to significantly reduced error rates. Intuitively, the perceptron updates on individual examples can be viewed as a form of stochastic gradient descent. Parameter averaging smoothes out the randomness in this process. Fig. 2 illustrates this intuition graphically. The figure visualizes the evolution of the parameters Φ_{sc} during training by projecting them onto their first two principal components. Note how the averaged parameter estimates "spiral" down more quickly to the final solution.

As mentioned in section 2.5, for perceptron training with the matrix factorization in eq. (14), there are two possible averaging procedures. The results generally show that better performance is obtained by optimizing the matrices Λ_{sc} while averaging the matrices Φ_{sc}. We can offer one possible intuition for this trend. As noted earlier, the factorization in eq. (14) is not unique. Therefore we can imagine a sequence of parameter estimates that involve different values for Λ_{sc} but equal values for Φ_{sc}. (That is, the varying estimates for Λ_{sc} differ only by a unitary transformation.) In this case, the constant value of Φ_{sc} will be returned by averaging the matrices Φ_{sc} using eq. (13), but not by averaging the matrices Λ_{sc} using eq. (15). Though this is a contrived scenario unlikely to occur in practice, it suggests that averaging in Λ_{sc} can lead to nonsensical results.

Table 4. Frame error rates from different forms of parameter averaging: no averaging, averaging in Φ by eq. (16), and averaging in Λ by eq. (17). See text for details.

#	Frame Error Rate (%)		
mix	no averaging	averaging in Φ	averaging in Λ
1	38.6	30.0	37.2
2	34.9	27.6	30.9
4	32.5	26.0	25.5
8	30.6	26.5	26.6

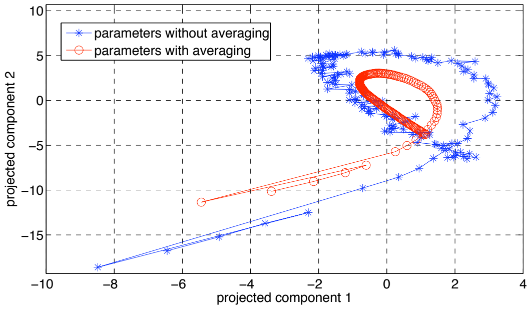

Figure 2. The trajectory of CD-HMM parameters during training. The figure visualizes the parameters Φ_{sc} by projecting them onto their first two principal components. Parameter averaging leads to faster convergence.

3.5. Benefits of initialization

For perceptron training in discrete HMMs, parameter values can simply be initialized as zeroes (Collins, 2002). However, in CD-HMMs, the discriminant function is not generally a linear function of the parameters, and the required optimization is not convex. Thus, depending on the quality of the initialization, the potential exists to get trapped in local minima.

Table 3.5 compares perceptron training with two different initializations: essentially, zero values versus maximum likelihood estimates. Interestingly, when we update the parameters Φ_{sc} using eq. (13), we observe no significant difference between these two initializations. On the other hand, when we update the parameters Λ_{sc} using eq. (15), we observe that much better performance is observed by initializing with maximum likelihood estimates. These results suggest that the much faster convergence from the matrix factorization in eq. (14) comes at the expense of creating a more treacherous optimization.

Table 5. Frame error rates from perceptron training with different initializations: zero values versus maximum likelihood (ML) estimates. The left results used the Φ-update in eq. (13); the right results used the Λ-update in eq. (15). See text for details.

#	Frame Error Rate (%)			
mix	$\Phi^{(0)}=0$	$\Phi^{(0)}=\Phi^{\mathrm{ML}}$	$\Lambda^{(0)}=0$	$\Lambda^{(0)}=\Lambda^{\mathrm{ML}}$
1	32.5	32.2	32.0	30.0
2	32.3	31.5	32.6	27.6
4	30.2	30.7	34.3	26.0
8	29.4	30.4	34.0	26.5

4. Discussion

In this paper, we have explored various matrix updates for perceptron training of CD-HMMs. As our main contributions, we analyzed numerous issues of parameterization and smoothing that do not arise in perceptron training of discrete HMMs; we also performed systematic experiments in ASR to understand how these issues play out in practice. Our results show that not all forms of discriminative training are equally effective: indeed, matrix reparameterizations and factorizations can have a significant effect on classification performance as well as rates of convergence. Future work will concentrate on applying the best practices identified here to larger scale problems in ASR, as well as exploring low-rank factorizations of eq. (14) that can be viewed as a form of dimensionality reduction.

Acknowledgements

This work was supported by NSF Award 0812576. Fei Sha is partially supported by the Charles Lee Powell Foundation. We thank the reviewers for many useful comments.

References

Bahl, L. R., Brown, P. F., de Souza, P. V., & Mercer, R. L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. *Proc. of International Conference of Acoustic, Speech and Signal Processing (ICASSP)* (pp. 49–52). Tokyo.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, vol. 20, 161–168. MIT Press.

Bottou, L., & LeCun, Y. (2004). Large scale online learning. In *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proc. of Empirical Methods in Natural Language Processing (EMNLP 2002).*

Freund, Y., & Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning* (pp. 277–296).

Gentile, C. (2002). A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res.*, *2*, 213–242.

Huang, X., Acero, A., & Hon, H.-W. (2001). *Spoken language processing*. Prentice-Hall.

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

Juang, B.-H., & Katagiri, S. (1992). Discriminative learning for minimum error classification. *IEEE Trans. Sig. Proc.*, *40*, 3043–3054.

Lamel, L. F., Kassel, R. H., & Seneff, S. (1986). Speech database development: design and analysis of the acoustic-phonetic corpus. *Proceedings of the DARPA Speech Recognition Workshop* (pp. 100–109).

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

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

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

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*.

Sha, F., & Saul, L. K. (2009). Large margin training of continuous density hidden markov models. In J. Keshet and S. Bengio (Eds.), *Automatic speech and speaker recognition: Large margin and kernel methods*. Wiley-Blackwell.

Woodland, P. C., & Povey, D. (2000). Large scale discriminative training for speech recognition. *Proc. of Automatic Speech Recognition (ASR2000).*

Yu, D., Deng, L., He, X., & Acero, A. (2007). Large-margin minimum classification error training for large-scale speech recognition tasks. *Prof. of International Conference on Acoustic, Speech and Signal Processing*.