Online Learning and Acoustic Feature Adaptation in Large Margin Hidden Markov Models

Chih-Chieh Cheng*, Fei Sha and Lawrence K. Saul

***Abstract*—We explore the use of sequential, mistake-driven updates for online learning and acoustic feature adaptation in large margin hidden Markov models (HMMs). The updates are applied to the parameters of acoustic models after the decoding of individual training utterances. For large margin training, the updates attempt to separate the log-likelihoods of correct and incorrect transcriptions by an amount proportional to their Hamming distance. For acoustic feature adaptation, the updates attempt to improve recognition by linearly transforming the features computed by the front end. We evaluate acoustic models trained in this way on the TIMIT speech database. We find that online updates for large margin training not only converge faster than analogous batch optimizations, but also yield lower phone error rates than approaches that do not attempt to enforce a large margin. Finally, experimenting with different schemes for initialization and parameter-tying, we find that acoustic feature adaptation leads to further improvements beyond the already significant gains achieved by large margin training.**

I. Introduction

Most existing systems for automatic speech recognition (ASR) are based on continuous-density hidden Markov models (CD-HMMs), whose parameters must be estimated from large training corpora of transcribed speech [1]. The simplest approach to this problem is maximum likelihood (ML) estimation, which attempts to maximize the joint likelihood of the training data. However, ML estimation has well-known limitations for ASR. At best, CD-HMMs provide only an approximate model of the tremendous variability observed in real speech. When such models are estimated from training data, improvements in their joint likelihoods do not always translate into fewer recognition errors. This realization has led researchers to develop other objective functions for parameter estimation that more closely track the error rate (however it is measured).

A great deal of research in ASR has focused on discriminative training of HMMs [2], [3], [4]. Perhaps the most popular framework for discriminative training is maximum mutual information (MMI) estimation (MMI). In this framework, model parameters are estimated to maximize the mutual information between the desired recognizer output and the acoustic features computed by the front end. More recently, building on the successes of support vector machines [5], [6] and various extensions thereof [7], [8] in the machine learning community, a number of researchers in ASR have also explored large margin methods for discriminative training of HMMs [9], [10], [11], [12].

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 ML estimation. Large corpora are typically managed by parallelizing batch computations of parameter updates across many different nodes, then combining the individual results to average over training utterances. For large-vocabulary ASR, discriminative training can also be accelerated by using lattices to provide a compact representation of alternative hypotheses [13]. Nevertheless, the scaling of discriminative methods to large-scale problems remains an important area 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 [14], [15]. 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 perform better than their batch counterparts.

Motivated by the potential of this approach for ASR, in this paper we investigate an online algorithm for discriminative training of HMMs. The algorithm optimizes the parameters of acoustic models in an incremental fashion, updating them after the decoding of each training utterance. The first main contribution of our paper is to propose a particular reparameterization of acoustic models that lends itself very well to this type of training. We present experimental results for acoustic models trained in this way on the TIMIT speech corpus [16]. The TIMIT corpus is a small-scale but still widely used benchmark [17], [18], [19], [20] for evaluating new approaches to hidden Markov modeling in ASR. We systematically compare the effects of different parameterizations, initializations, and averaging schemes on convergence rates and phone recognition accuracies. Our results illustrate a set of best practices for online, discriminative training that yield the most consistently significant and rapid reductions in phone recognition error rates [21].

Chih-Chieh Cheng and Lawrence K. Saul are with Department of Computer Science and Engineering, University of California, San Diego. e-mail: (chc028@cs.ucsd.edu, saul@cs.ucsd.edu)

Fei Sha is with Department of Computer Science, University of Southern California. e-mail: (feisha@usc.edu)

The second main contribution of our paper is to investigate online updates for large margin training of HMMs [9], [22], [23], [10], [11], [24], [25]. The goal of large margin training is to assign significantly higher scores to correct transcriptions than competing ones; in particular, the margin between these scores is required to grow in proportion to the total number of recognition errors [22], [23], [24], [25], [8]. Empirically, large margin training has improved the performance of many systems beyond other leading discriminative approaches. We propose online updates that incrementally adapt the model parameters after a margin-based decoding of each training utterance. Comparing online versus batch implementations of large margin training, we find that the online methods converge more quickly. We also find that they yield acoustic models with better performance on phone recognition than other approaches—both online and batch—that do not attempt to enforce a large margin [26].

The third main contribution of our paper is to study online updates that simultaneously transform the acoustic feature space used for ASR. Specifically, we show how to adapt the acoustic features computed by typical front ends in order to increase the margin of correct recognition. We adapt the feature space by learning highly discriminative linear projections of acoustic features concatenated from multiple adjacent analysis windows. Optimizing the acoustic features in the front end presents new challenges for online learning. First, the optimization landscape becomes considerably more complex. Second, the projection matrix appears to be especially sensitive to the choice of learning rates. To deal with these difficulties, we explore many different schemes for initialization and parameter-tying [27], [28]. Interestingly, our best results are obtained by training several recognizers in parallel while tying the projection matrix used to compute acoustic features in their front ends. In our experiments, this form of parameter-tying *across different recognizers* yields consistent improvement beyond the already significant gains of large margin training.

We have published our preliminary explorations of these ideas in three previous studies [21], [26], [29]. In this paper, we provide a unified presentation of these ideas and also include additional experiments on larger model sizes and different training paradigms.

The organization of this paper is as follows. In section II, we review standard approaches to acoustic modeling in ASR, as well as useful performance metrics. In section III, we introduce our online updates for discriminative training of HMMs. In sections IV and V, we extend these updates to incorporate large margin constraints and to adapt the acoustic feature space. In each of these sections, we also present experimental results on the TIMIT speech corpus; by interleaving results in this way, we hope to convey the evolution of ideas and practices that guided our own investigations. Finally, in section VI, we summarize our most important findings and discuss future directions for research.

II. Background

We begin by reviewing basic notation for CD-HMMs and popular methods for parameter estimation. We also report several previous benchmarks for acoustic modeling, against which subsequent models will be judged.

A. Continuous-density hidden Markov models

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 $\mathcal{P}(s_1)$, the hidden state transition matrix $\mathcal{P}(s_{t+1}|s_t)$, and the emission densities $\mathcal{P}(x_t|s_t)$. In terms of these quantities, the joint distribution is given by:

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

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 parameterized by Gaussian mixture models (GMMs), with mixture weights $\mathcal{P}(c|s)$, mean vectors μ_{sc}, and covariance matrices Σ_{sc} for the c^{th} component of each hidden state. In terms of these parameters, the emission densities are given by:

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

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

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

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 posterior probability.

B. Generative versus discriminative approaches

The simplest form of training for CD-HMMs is ML estimation. ML estimation is based on viewing CD-HMMs as a generative model of speech. 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:

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

Once the overall model architecture is specified, maximum-likelihood 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, and does not involve tuning parameters such as learning rates. All these properties make it very attractive as a starting point for ASR.

However, ML estimation has one serious drawback: increasing the joint likelihood in eq. (4) does not generally decrease the error rate of the recognizer. Recent empirical [30] and theoretical [31] analyses have highlighted the shortcomings of ML estimation when the estimated models are not perfectly

matched to the data. To overcome these shortcomings, we must consider discriminative methods for parameter estimation in CD-HMMs.

Discriminative training of CD-HMMs has a long history in ASR [2], [3], [4], and new work continues to appear in this area. The fundamental idea behind discriminative training is to seek parameters that minimize the error rate rather than attempting to model the data itself. One popular method for discriminative training in ASR is based on maximizing the mutual information (MMI) between observations and states:

$$\Theta_{\mathrm{MMI}} = \mathrm{argmax}_{\Theta} \sum_{n=1}^{N} \log \frac{\mathcal{P}(\mathbf{x}_n, \mathbf{y}_n|\Theta)}{\mathcal{P}(\mathbf{x}_n|\Theta)\mathcal{P}(\mathbf{y}_n|\Theta)}. \quad (5)$$

The maximization is typically done by gradient ascent or extended Baum-Welch updates; both methods require computing derivatives of the right hand side with respect to the parameter Θ. Closely related to MMI is conditional maximum likelihood (CML) estimation:

$$\Theta_{\mathrm{CML}} = \mathrm{argmax}_{\Theta} \sum_{n=1}^{N} \log \mathcal{P}(\mathbf{y}_n|\mathbf{x}_n, \Theta). \quad (6)$$

CML differs only slightly from MMI in its estimation of transition probabilities and language model parameters; the methods are equivalent if these parameters are not updated. Another popular discriminative method is minimum classification error (MCE), which directly minimizes the number of sequence misclassifications. In MCE, the parameters are found by optimizing:

$$\Theta_{\mathrm{MCE}} = \mathrm{argmax}_{\Theta} \sum_{n=1}^{N} \mathrm{sign}\left[\max_{s \neq y_n} \log \frac{\mathcal{P}(\mathbf{x}_n, \mathbf{s}|\Theta)}{\mathcal{P}(\mathbf{x}_n, \mathbf{y}_n|\Theta)}\right], \quad (7)$$

where $\mathrm{sign}[z] = 1$ if $z > 0$ and $\mathrm{sign}[z] = 0$ if $z \leq 0$. Since the right hand side of eq. (7) is nondifferentiable, MCE usually replaces the *max* function by a *softmax* function, then optimizes the parameters by gradient ascent.

Discriminative training of CD-HMMs is more complicated than ML estimation for several reasons: (i) probabilities (and their gradients) must be computed not only for desired state sequences, but also for competing ones; (ii) many update rules involve learning rates, which must be carefully tuned; (iii) convergence is not as fast or as stable as the EM algorithm for ML estimation.

C. Existing benchmarks

The benefits of discriminative training have been demonstrated in many different tasks and applications. We used the TIMIT speech corpus [16] to evaluate the competing models discussed in this paper. The speech signals in this corpus have been manually segmented and aligned with their phonetic transcriptions. These transcriptions provide ground-truth labels for benchmarking different acoustic models on the problem of phone recognition.

We adopted the same methodology as recent benchmarks on this data set [12]. For the front end, we computed 39-dimensional acoustic feature vectors of mel-frequency cepstral coefficients on sliding windows of speech. For each utterance, we performed cepstral mean subtraction, but not endpointing. For the back end, we trained CD-HMMs using the manually aligned phonetic transcriptions as target hidden states. We did not introduce or optimize word insertion probabilities to lower the frame and phone error rates. The CD-HMMs had 48 hidden states (one per context-independent phone) and GMMs that varied in size from one to 128 mixture components. We used the standard partition of the TIMIT corpus, yielding roughly 1.1 million, 120K, and 57K frames respectively for training, test, and holdout data. This standard partition corresponds to 5 hours of training utterances and 30 minutes of test utterances.

TABLE I
PHONE ERROR RATES FOR CD-HMMS OF VARYING SIZE ON THE TIMIT SPEECH CORPUS, AS OBTAINED BY MAXIMUM LIKELIHOOD (ML), CONDITIONAL MAXIMUM LIKELIHOOD (CML), AND MINIMUM CLASSIFICATION ERROR (MCE) ESTIMATION. THE RESULTS IN THE FIRST FOUR ROWS ARE FROM PREVIOUS BENCHMARKS [12]. THE LEFT COLUMN SHOWS THE NUMBER OF MIXTURE COMPONENTS PER GMM.

# mixture	Phone Error Rate (%)		
component	ML	CML	MCE
1	41.5	36.4	35.6
2	38.0	34.6	34.5
4	34.9	32.8	32.4
8	32.3	31.5	30.9
16	30.8		
32	31.8		
64	33.4		
128	35.9		

We measured performance by comparing the hidden state sequences inferred by Viterbi decoding of CD-HMMs to the phonetic transcriptions provided by the TIMIT corpus. In calculating error rates, we followed the standard practice of mapping 48 phonetic classes down to 39 broader categories [32]. In general, 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 [32]. The phone error rate provides the more relevant metric for ASR. However, in some instances, we also report the frame error rate because it is more directly minimized by the algorithms we study in later sections.

Table I presents previous benchmarks [12] on the TIMIT speech corpus, as well as some additional results on larger models trained by ML. The table shows the phone error rates of CD-HMMs trained by ML, CML, and MCE. Note that discriminative training leads to significantly lower error rates than ML estimation for models of the same size. The results in Table I will provide useful baselines for the models we discuss in subsequent sections. Also, except where otherwise noted, the ML models in Table I were used to initialize all discriminatively trained models mentioned in this paper.

III. ONLINE UPDATES FOR HMMS

As discussed in the previous section, discriminative training of acoustic models is considerably more expensive than ML estimation. Thus it is useful to consider procedures that simplify or accelerate discriminative training. In this section, we explore the potential of online updates to achieve these goals.

Our approach is motivated by one of the simplest and oldest algorithms for online learning: namely, the perceptron [33]. Perceptrons use a mistake-driven update rule to learn linear decision boundaries between classes of positively and negatively labeled examples. An exciting line of recent work has generalized the perceptron algorithm to discriminative training of discrete HMMs [34]. The perceptron algorithm for discrete HMMs combines simple additive updates with Viterbi decoding of training examples. On problems in part-of-speech tagging [35] and base noun phrase chunking [36], this algorithm outperformed other leading approaches to discriminative training. We seek to replicate these successes in HMMs for ASR.

New difficulties arise in perceptron-like training of CD-HMMs that are not present in discrete HMMs. These difficulties are rooted in the parameterization of emission densities. For example, in CD-HMMs, online updates must adapt the means and covariance matrices of multivariate Gaussian distributions. However, simple, additive updates can violate the positive definiteness of covariance matrices, thus requiring further computation to maintain these constraints.

This section is organized as follows. In section III-A, we describe the perceptron algorithm for HMMs in general terms. In sections III-B through III-D, we consider how to parameterize the acoustic models used in ASR for this type of training and discuss various issues that arise from different parameterizations. Finally, in section III-E, we present experimental results on the TIMIT speech corpus. Our results highlight the parameterizations of acoustic models that yield the most consistent and rapid reductions in phone error rates.

A. Mistake-driven updates

Perceptron training in HMMs is based on a so-called discriminant function over observation and hidden state sequences:

$$\mathcal{D}(\mathbf{x}, \mathbf{s}) = \log \mathcal{P}(s_1) + \sum_{t>1} \log \mathcal{P}(s_t|s_{t-1}) + \sum_t \log \mathcal{P}(x_t|s_t). \tag{8}$$

The discriminant function is essentially the logarithm of the joint probability distribution in eq. (1). For simplicity, we have suppressed the dependence of the discriminant function on the parameters Θ of the CD-HMM. In terms of the discriminant function, a target state sequence $\mathbf{y}$ will be correctly inferred from the observation sequence $\mathbf{x}$ if:

$$\forall \mathbf{s} \neq \mathbf{y}, \quad \mathcal{D}(\mathbf{x}, \mathbf{y}) > \mathcal{D}(\mathbf{x}, \mathbf{s}). \tag{9}$$

Note that eq. (9) defines a set of inequalities for all incorrect transcriptions $\mathbf{s} \neq \mathbf{y}$. In our experiments, the target state sequences are the manually aligned phonetic transcriptions in the TIMIT corpus. In this work, we make the simplifying assumption that there is a unique target state sequence (as opposed to considering multiple valid target sequences that decode to the same word sequence).

In general, it is not possible for a model to satisfy all the constraints in eq. (9). We use the following loss function [37] to measure the total constraint violation across the entire training corpus:

$$\mathcal{L}(\Theta) \;=\; \sum_n \left[\max_{\mathbf{s} \neq \mathbf{y}_n} \mathcal{D}(\mathbf{x}_n, \mathbf{s}) - \mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) \right]^+, \tag{10}$$

where $[z]^+ = \max(z, 0)$ indicates the nonnegative hinge function. The right hand side of eq. (10) computes a weighted count of the training utterances that do not satisfy the constraints in eq. (9). In particular, each incorrectly decoded utterance is weighted by the log-likelihood gap between the correct transcription and the Viterbi decoding, as computed by eq. (3).

To minimize the loss function in eq. (10), we consider the online, mistake-driven update:

$$\Theta \leftarrow \Theta + \eta \frac{\partial}{\partial \Theta} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*) \right], \tag{11}$$

where $\eta > 0$ is a carefully chosen learning rate. Note that the update in eq. (11) is only applied when Viterbi decoding returns an incorrect transcription $\mathbf{s}_n^* \neq \mathbf{y}_n$. The update can be viewed as a form of stochastic gradient descent [38] on the loss function in eq. (10), which has also been studied in the related context of graph transformer networks [39]. The gradient in eq. (11) computes the fastest search direction in parameter space to minimize the log-likelihood gap between target and inferred state sequences. For discriminative training, we may also adapt the parameters of acoustic models 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.

Perceptron training updates parameters in a sequential manner, looping through all the training examples until either the algorithm converges or no longer reduces the average number of classification errors. We follow a similar procedure for updating the parameters of acoustic models for ASR. In general, mistake-driven updates 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 parameters from perceptron training across different updates of the training examples [40], [41]. In practice, this sort of averaging reduces the noise in the parameter vector by damping fluctuations in the decision boundary that occur during training. Let $\Theta^{(j)}$ represent the parameter estimates after the perceptron update in eq. (11) has been applied for the j^{th} time. We compute the averaged parameters $\tilde{\Theta}^{(r)}$ after r updates as:

$$\tilde{\Theta}^{(r)} = \frac{1}{r} \sum_{j=1}^{r} \Theta^{(j)}. \tag{12}$$

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. In addition to parameter averaging, convergence may also be obtained by decreasing the learning rate over time; however, experimenting with this strategy, we found it to be much less effective than parameter-averaging.

B. Parameterization of GMMs

Conventionally, CD-HMMs are parameterized in terms of transition matrices and emission densities. The choice of parameterization plays an important role in online learning. 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 these parameters may not preserve their nonnegativity, which is necessary for the discriminant function in eq. (8) 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 online learning, as well as the nature of the averaging in eq. (12).

In the rest of this section, we flesh out these issues, concentrating mainly on the parameterization of the GMMs. In general, the GMM parameters in HMMs play a much more important role than the transition probabilities; moreover, the latter 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. However, the most straightforward online updates are given in terms of the log-mixture weights $\nu_{sc} = \log \mathcal{P}(c|s)$ and inverse covariance matrices Σ_{sc}^{-1}. The mixture weights are best updated in the log domain to ensure that they remain nonnegative. It is also simpler to compute the derivatives in the update rule, eq. (11), 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 online updates of the form:

$$\nu_{sc} \leftarrow \nu_{sc} + \eta \frac{\partial}{\partial \nu_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right], \quad (13)$$

$$\mu_{sc} \leftarrow \mu_{sc} + \eta \frac{\partial}{\partial \mu_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right], \quad (14)$$

$$\Sigma_{sc}^{-1} \leftarrow \Sigma_{sc}^{-1} + \eta \frac{\partial}{\partial \Sigma_{sc}^{-1}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right]. \quad (15)$$

The last update in eq. (15) 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, computed from eq. (2), remain normalizable and finite.

Though simple in concept, the stochastic gradient descent in eqs. (13–15) may require the careful tuning of multiple learning rates in order to succeed. Alternatively, a common strategy is to only optimize the mean parameters of GMMs.

C. Reparameterization of GMMs

Building on ideas from previous work [12], we investigate a reparameterization of GMMs that aggregates the mixture weight, mean, and covariance matrix parameters associated with each Gaussian mixture component into a single augmented matrix. Let

$$\gamma_{sc} = -\log\left(\frac{\mathcal{P}(c|s)}{\sqrt{(2\pi)^d |\Sigma_{sc}|}}\right) \quad (16)$$

denote the log of the scalar prefactor that weights each Gaussian mixture component. Then for each Gaussian mixture component, consider the matrix:

$$\Phi_{sc} = \begin{bmatrix} \Sigma_{sc}^{-1} & -\Sigma_{sc}^{-1}\mu_{sc} \\ -\mu_{sc}^\top \Sigma_{sc}^{-1} & \mu_{sc}^\top \Sigma_{sc}^{-1} \mu_{sc} + \gamma_{sc} \end{bmatrix}. \quad (17)$$

In eq. (17), 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:

$$\mathcal{P}(x|s) = \sum_c e^{-\frac{1}{2} z^\top \Phi_{sc} z} \quad \text{where} \quad z = \begin{bmatrix} x \\ 1 \end{bmatrix}. \quad (18)$$

We can use the reparameterization in eqs. (17–18) to adapt the matrices Φ_{sc} by mistake-driven updates, as opposed to the GMM parameters in the previous section. In this way, we can replace the three separate updates in eqs. (13–15) by the single update:

$$\Phi_{sc} \leftarrow \Phi_{sc} + \eta \frac{\partial}{\partial \Phi_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right]. \quad (19)$$

Our experiments in section III-E compare the performance of this single update to the separate updates in eqs. (13–15).

We impose a further constraint on the matrices Φ_{sc} that is not immediately implied by the reparameterization in eq. (17); namely, we require them to be positive semidefinite. Although the inverse covariance matrices Σ_{sc}^{-1} are constrained to be positive definite, the matrices Φ_{sc} in eq. (17) do not inherit this property if the scalar prefactors γ_{sc} are defined from existing GMMs as in eq. (16). Does this constraint limit the representational capacity of our acoustic models? Naively, a positive semidefinite constraint $\Phi_{sc} \succeq \mathbf{0}$ appears to suggest that the probability density $P(x|s)$ in eq. (18) cannot be arbitrarily large—that is, it cannot be arbitrarily peaked or concentrated about its mean value.

In fact, the positive semidefinite constraints on Φ_{sc} do not limit the representational capacity of our acoustic models. The capacity is preserved by relaxing the constraint that these models define properly normalized continuous densities over the acoustic feature space. Note that in CD-HMMs, the Viterbi sequences are determined by the likelihood ratios of emission densities in different states. Given any CD-HMMs, with arbitrarily peaked emission densities, consider the unnormalized CD-HMM whose emission densities are multiplied by a constant factor across all states. The likelihood ratios between states are unchanged. However, if the multiplicative factor is sufficiently large, then all the likelihood ratios can be preserved by the reparameterization in eqs. (17–18), provided that eq. (16) incorporates an additive offset from the logarithm of the multiplicative factor. In this way, the reparameterized (unnormalized) acoustic models in this section can be initialized to replicate the exact decoding procedures of any CD-HMM.

Note that like the earlier update in eq. (15) for the inverse covariance matrix, the update in eq. (19) 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.

Unlike the earlier update in eq. (15) 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. (18) is perfectly well defined for all positive semidefinite matrices $\Phi_{sc} \succeq 0$. Thus the online update in eq. (19) can learn to use unnormalized Gaussians with unbounded (though nonnegative) variance if they do indeed lead to fewer classification errors. Essentially, zero eigenvalues in the matrices Φ_{sc} indicate directions (perhaps invariances) in feature space that are not useful for large margin classification. However, we do not allow the matrices Φ_{sc} to have negative eigenvalues; otherwise, observations would be more likely to be associated with particular states even as they deviated further away from the centroids of those states.

We emphasize again that the update in eq. (19) effectively removes the constraint that the Gaussian distributions are properly normalized. Note that for a properly normalized Gaussian distribution, the bottom diagonal matrix element of Φ_{sc} in eq. (17) 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. The resulting model does not define a proper density over acoustic feature vectors; however, it uses the same decoding procedures (based on dynamic programming) as CD-HMMs.

D. Matrix factorizations

The update in eq. (19) 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 that were relaxed 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 the state and mixture component's centroid 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 converge more slowly than unconstrained methods, particularly when the projection and gradient steps work at cross purposes.

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

$$\Phi_{sc} = \Lambda_{sc}\Lambda_{sc}^T \tag{20}$$

makes explicit that the matrix Φ_{sc} is positive semidefinite. With this factorization, we can replace the update in eq. (19) by stochastic gradient descent in the matrix Λ_{sc}:

$$\Lambda_{sc} \leftarrow \Lambda_{sc} + \eta \frac{\partial}{\partial \Lambda_{sc}} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \mathbf{s}_n^*)\right]. \tag{21}$$

Note that in this update, the square matrices Λ_{sc} (of the same size as Φ_{sc}) are completely unconstrained.

The update in eq. (21) 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. (21): the discriminant function is neither linear nor convex in the matrices Λ_{sc}. Our experiments in section III-E2 attempt to determine which potential advantages and disadvantages of this matrix factorization are realized in practice.

We note that the factorization in eq. (20) is not unique. While a matrix square root satisfying eq. (20) 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. We were curious whether such a factorization would accelerate learning (because a lower triangular matrix has fewer parameters to estimate than a full matrix) or decelerate learning (because optimizations sometimes converge more quickly in an enlarged parameter space). Since the optimization is non-convex, we were also curious whether such a factorization might provide a consistently better initialization. In section III-E2, 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. (20) raises another issue related to the averaging of parameter estimates as in eq. (12). For training, we can update the matrices Φ_{sc} directly by eq. (19) or indirectly by eq. (21). 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. (12), one approach is to average the parameter estimates for Φ_{sc} as:

$$\tilde{\Phi}_{sc}^{(r)} = \frac{1}{r}\sum_{j=1}^{r} \Phi_{sc}^{(j)} = \frac{1}{r}\sum_{j=1}^{r} \Lambda_{sc}^{(j)}\Lambda_{sc}^{(j)\top}. \tag{22}$$

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

$$\tilde{\Phi}_{sc}^{(r)} = \tilde{\Lambda}_{sc}^{(r)}\tilde{\Lambda}_{sc}^{(r)\top}, \quad \text{where } \tilde{\Lambda}_{sc}^{(r)} = \frac{1}{r}\sum_{j=1}^{r} \Lambda_{sc}^{(j)}. \tag{23}$$

In section III-E4, we evaluate and present results for both types of averaging.

E. Experiments

We experimented on the TIMIT speech corpus (see section II-C) to evaluate the online updates described in sec-

tions III-A–III-D. Our experiments were designed not only to assess the potential benefits of discriminative training, but also to compare different mistake-driven updates for online learning of HMMs.

Online updating of acoustic models for ASR 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 ML estimation (section III-B), or by aggregating the parameters for each mixture component into a single matrix (section III-C)? (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 III-D)? (iii) in which parameter space should we average to obtain smoothed parameter estimates for testing (section III-D)? Our experimental results provide fairly definitive answers to these questions.

Before presenting the results, we briefly discuss our methodology. We examined test error rates across a wide range of model sizes by varying the number of Gaussian mixture components per hidden state. We report these results for acoustic models of different sizes to illustrate various general trends. In practice, however, the correct model size is not known in advance; it is a hyperparameter that must be determined by the performance on held-out data. Thus, in each table of results that follows, we also indicate in **boldface** the test error rate of the model that had the lowest phone error rate on the TIMIT development set. The model selected in this way was often though not always the best model on the test set.

1) Overall benefits of online learning: We begin by reporting results that confirm the well-known benefits of discriminative training and online learning. Table II compares the best-performing CD-HMMs obtained by ML estimation to the best performing acoustic models obtained by online, mistake-driven updates. The latter used the matrix update in eqs. (20–21) and the averaging scheme in eq. (22). The results show that the online updates lead to significant reduction in both frame and phone error rates for models with up to sixteen Gaussian mixture components per hidden state. The improvements in frame error rates are larger than the improvements in phone error rates; this discrepancy reflects the fact that the discriminant function more closely tracks the Hamming distance (not the edit distance) between target and Viterbi phone sequences. For reference, Table II also shows previously published benchmarks [12] from the two most popular batch approaches to discriminative training. It is interesting that for all model sizes, the online updates outperform these batch approaches.

Though training times vary from experiment to experiment, we observed the following general trend. For the smallest models (e.g., 1-2 mixture components per hidden state), the discriminative training took much longer than the initial ML estimation; for medium-sized models (e.g., 4-8 mixture components per hidden state), the discriminative training took roughly the same amount of time; finally, for the largest models (e.g., 16-32 mixture components per hidden state), the discriminative training took less time than the initial ML estimation. It seems that the speed-ups from online learning are most pronounced for large model sizes; in particular, in this regime, the speed-ups from stochastic gradient descent appear to more than offset the extra computations (e.g., Viterbi decoding of training utterances) required for discriminative training.

TABLE II

FRAME AND PHONE ERROR RATES FOR ACOUSTIC MODELS OF VARYING SIZE, AS OBTAINED BY MAXIMUM LIKELIHOOD (ML) ESTIMATION, ONLINE MISTAKE-DRIVEN UPDATES, 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 [12]. THE LEFT COLUMN SHOWS THE NUMBER OF MIXTURE COMPONENTS PER GMM.

#	Frame Error Rate (%)		Phone Error Rate (%)			
mix	ML	online	ML	online	CML	MCE
1	39.7	31.4	41.5	33.6	36.4	35.6
2	36.2	30.1	38.0	32.3	34.6	34.5
4	33.1	29.5	34.9	31.4	32.8	32.4
8	30.7	28.8	32.3	30.1	31.5	30.9
16	29.5	**28.6**	30.8	**29.7**		
32	29.9	29.3	31.8	30.9		

TABLE III

FRAME ERROR RATES FROM DISCRIMINATIVE TRAINING OF ACOUSTIC MODELS WITH DIFFERENT FORMS OF STOCHASTIC GRADIENT DESCENT: UPDATING (ν, μ, Σ^{-1}) IN EQS. (13-15) VERSUS UPDATING Φ IN EQS. (17–19).

#	Frame Error Rate (%)	
mix	(ν, μ, Σ^{-1})	Φ
1	37.0	32.2
2	36.5	31.5
4	35.8	31.0
8	33.9	30.9
16	31.8	30.5
32	**30.1**	**30.4**

2) Benefits of reparameterization: As noted in section III-B, for the conventional parameters of GMMs, it is often necessary to tune separate learning rates for discriminative training to succeed. Our experiments bore out these difficulties. The left column of Table III shows our best results from discriminative training with the conventional parameterization of GMMs. In fact, these results were obtained by updating just the mixture weights and mean vectors of the GMMs; 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 practice, most of the performance gains in discriminative training have been realized by optimizing the mean parameters in GMMs.

The reparameterization in section III-C greatly simplifies both the form of the discriminant function, eq. (18), and the resulting online updates. The results in the rightmost column of Table III reveal the benefits of this approach. These results were obtained using the reparameterization in eq. (17), the update in eq. (19), and the averaging in eq. (12). Note that mistake-driven updates based on the parameters Φ_{sc} from eq. (17) leads to significantly lower frame error rates across all model sizes.

3) Benefits of matrix factorization: We also experimented with the matrix factorization in eq. (20). Updating the matrices Λ_{sc} by eq. (21) led to the results shown in Table IV. The

TABLE IV

FRAME ERROR RATES FROM THE UPDATE IN EQ. (19) VERSUS THE UPDATE IN EQ. (21). 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.

# mix	Frame Error Rate (%)					
	Φ		Λ-SVD		Λ-Cholesky	
1	32.2	(243)	31.4	(32)	35.5	(149)
2	31.5	(258)	30.1	(37)	35.6	(61)
4	31.0	(296)	29.5	(6)	32.3	(2)
8	30.9	(131)	28.8	(2)	31.4	(2)
16	30.5	(7)	**28.6**	(3)	**29.0**	(2)
32	**30.4**	(2)	29.3	(3)	29.6	(3)

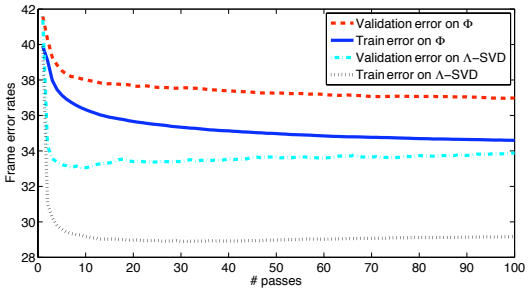

Fig. 1. Comparison of online, mistake-driven updates with and without the matrix factorization in eq. (20). See text for details.

TABLE V

FRAME ERROR RATES FROM DIFFERENT FORMS OF PARAMETER AVERAGING: NO AVERAGING, AVERAGING IN Φ BY EQ. (22), AND AVERAGING IN Λ BY EQ. (23). SEE TEXT FOR DETAILS.

# mix	Frame Error Rate (%)		
	no averaging	averaging in Φ	averaging in Λ
1	41.9	31.4	31.6
2	37.1	30.1	30.9
4	35.2	29.5	30.2
8	35.2	28.8	28.8
16	**33.5**	**28.6**	**28.4**
32	33.9	29.3	29.6

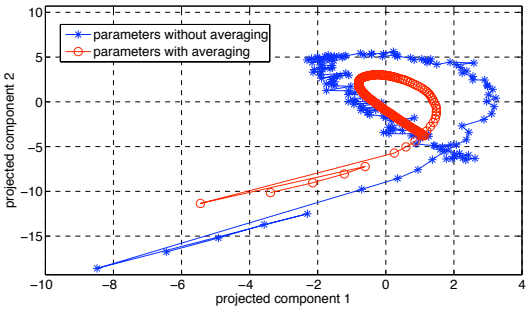

Fig. 2. The trajectory of CD-HMM parameters during training. The figure visualizes the parameters Φ_{sc} by projecting them onto their first two principal components. Parameter averaging leads to faster convergence.

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 III for updating the matrices Φ_{sc} by eq. (19). Note how the unconstrained factorization in eq. (20) leads to consistent further improvements beyond those obtained by the reparameterization in eq. (17). The factorization also leads to much faster convergence as measured by the numbers 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 the online, mistake-driven updates using the matrix factorization in eq. (20). 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 IV. When updating the matrices Λ_{sc} using eq. (21), the training error drops rapidly, and the acoustic models appear to start overfitting after just a few sweeps through the training data. By contrast, when updating the matrices Φ_{sc} using eq. (19), the training and holdout error rates drop much more slowly.

4) Benefits of averaging: Parameter averaging is an effective technique for reducing the fluctuations inherent to online learning. Table V demonstrates the benefits of parameter averaging in the setting of acoustic modeling, where it often leads to significantly reduced error rates. Intuitively, we can view the online, mistake-driven updates on individual utterances as stochastic gradient descent on the overall loss function. Parameter averaging smoothes out the randomness in this process. Fig. 2 illustrates this intuition by visualizing how the GMM parameters across all states and mixture components evolve during training. In particular, for the purpose of visualization, the figure shows the two dimensional trajectory obtained by projecting these GMM parameters onto their first two principal components. Note how the averaged parameter estimates "spiral" down more quickly to the final solution.

As mentioned in section III-D, for online, mistake-driven updates with the matrix factorization in eq. (20), there are two possible averaging procedures. The results show that better performance is generally (though not always) 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. (20) 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. (19), but not by averaging the matrices Λ_{sc} using eq. (21). Though this is a contrived scenario unlikely to occur in practice, it suggests that averaging in Λ_{sc} can lead to nonsensical results.

5) Benefits of initialization: For perceptron training in discrete HMMs, parameter values can simply be initialized as zeroes [34]. However, when GMMs are used in acoustic models, 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 VI compares the results from online, mistake-driven

TABLE VI
FRAME ERROR RATES FROM DIFFERENTLY INITIALIZED SETS OF MODEL PARAMETERS, ONE SET WITH ZERO VALUES, THE OTHER WITH MAXIMUM LIKELIHOOD (ML) ESTIMATES. THE LEFT RESULTS USED THE Φ-UPDATE IN EQ. (19); THE RIGHT RESULTS USED THE Λ-UPDATE IN EQ. (21). 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	32.2	32.2	**33.1**	31.4
2	33.4	31.5	34.9	30.1
4	32.0	31.0	35.7	29.5
8	32.0	30.9	36.2	28.8
16	**31.6**	30.5	35.0	**28.6**
32	32.3	**30.4**	37.9	29.3

updates using two different sets of initial model parameters: one set with zero values, the other with ML estimates. Two trends are clear. First, the ML initialization generally leads to better performance, especially as the model size is increased. Second, the zero-valued initialization leads to *worsening* performance with increasing model size when we update the parameters Λ_{sc} using eq. (21). These results suggest that the much faster convergence from the matrix factorization in eq. (20) comes at the expense of creating a more treacherous optimization.

IV. ONLINE UPDATES FOR LARGE MARGIN HMMS

In this section, we extend the online updates from the previous section to incorporate large margin constraints. Large margin training of HMMs seeks not only to minimize the empirical error rate, but also to separate the scores of correct and incorrect transcriptions by the largest possible amount, thus achieving better generalization on unseen data [8], [7]. This idea has been independently investigated by many researchers in acoustic modeling and ASR [42], [9], [23], [10], [11]. Our main goal here is to investigate simple, online updates for large margin training of acoustic models.

A. Large margin training

Let $(\mathbf{x}, \mathbf{y})$ denote an observation sequence and its ground truth transcription. The essence of large margin training lies in the following observation: whereas for correct recognition we merely require the inequalities in eq. (9), for correct recognition *by a large margin*, we additionally require that

$$\forall \mathbf{s} \neq \mathbf{y}, \quad \mathcal{D}(\mathbf{x}, \mathbf{y}) > \mathcal{D}(\mathbf{x}, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}), \tag{24}$$

where $\mathcal{H}(\mathbf{s}, \mathbf{y})$ is the Hamming distance between two hidden state sequences of the same length, and $\rho > 0$ is a constant margin scaling factor. In other words, for large margin training, the score of the correct transcription should exceed the score of any incorrect transcription by an amount that grows in proportion to the number of recognition errors. We discuss other types of distance penalties in section VI.

We can use dynamic programming to compute the hidden state sequence that most egregiously violates the margin constraint in eq. (24). We use $\tilde{\mathbf{s}}^*$ to denote this hidden state sequence. From eq. (24), we have:

$$\tilde{\mathbf{s}}^* = \mathrm{argmax}_{\mathbf{s} \neq \mathbf{y}} \left[\mathcal{D}(\mathbf{x}, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}) \right]. \tag{25}$$

The right hand side of eq. (25) can be maximized by a simple variant of the standard Viterbi algorithm [37]. We emphasize that the margin-based decoding selects and penalizes incorrect sequences that are *close* in log-likelihood but *far away* in Hamming distance. Put another way, if two competing sequences have the same log-likelihood, then the margin-based decoding will select and penalize the one with more (frame-level) transcription errors.

To measure the total amount of constraint violation in eq. (24), we define the loss function:

$$\mathcal{L}(\Theta) \;=\; \sum_n \left[\max_{\mathbf{s} \neq \mathbf{y}} \left[\mathcal{D}(\mathbf{x}, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}) \right] - \mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) \right]^+, \tag{26}$$

analogous to the loss function in eq. (10). For online training of large margin HMMs, we consider the following update rule:

$$\Theta \leftarrow \Theta + \eta \frac{\partial}{\partial \Theta} \left[\mathcal{D}(\mathbf{x}_n, \mathbf{y}_n) - \mathcal{D}(\mathbf{x}_n, \tilde{\mathbf{s}}_n^*) \right]. \tag{27}$$

The update is applied whenever the margin-based decoding in eq. (25) yields a state sequence that violates the inequality in eq. (24). Eq. (27) differs from eq. (11) in one critical aspect: namely, we replace the usual Viterbi sequence in eq. (3) by the sequence from margin-based decoding in eq. (25). This substitution changes the nature of the optimization in an important way: even if an utterance is correctly decoded, eq. (27) may still update the model parameters. In particular, the parameters will be updated if there exists an incorrect decoding whose log-likelihood is not sufficiently well separated from that of the correct transcription.

Though the margin scaling factor ρ does not appear explicitly in eq. (27), it directly affects the computation of $\tilde{\mathbf{s}}_n^*$. In fact, our experiments will show that the subtle change in eq. (27) leads to profoundly different updates. To obtain smoother parameter estimates over time, the results from eq. (27) can also be averaged as in eq. (12). We performed this averaging in all of our experiments.

B. Experiments

Following the same experimental set-up as in previous sections, we sought to investigate the potential benefits of online updates for large margin training. All CD-HMMs were initialized by ML estimation. Starting from these baseline CD-HMMs, we then compared the performance of the different online updates in eq. (11) and (27). For these comparisons, we used the parameterization in eq. (20) and the averaging in eq. (12), since these choices yielded the best results for online updates without large margin constraints. For the margin-based update, the results of training depend on the margin scaling factor ρ. We experimented with a wide range of values for this scaling factor.

Table VII shows the results from the best models trained in this way. (For the margin-based results, we chose the scaling factor ρ that yielded the lowest phone error rates on the held-out development set.) The results show that online updating with margin-based decoding significantly reduces the frame and phone error rates across all model sizes. In general, the frame error rates improve more than the phone

TABLE VII
FRAME ERROR RATES (*top*) AND PHONE ERROR RATES (*bottom*) ON THE TIMIT TEST SET FOR ACOUSTIC MODELS OF VARYING SIZE, AS OBTAINED BY MAXIMUM LIKELIHOOD (ML) ESTIMATION, ONLINE UPDATES WITH STANDARD VITERBI DECODING, ONLINE UPDATES WITH MARGIN-BASED DECODING, AND A BATCH IMPLEMENTATION OF LARGE MARGIN TRAINING [12].

# mixture component	Frame Error Rate (%)			
	Maximum likelihood	Online w/o margin	Online w/ margin	Batch w/ margin
1	39.7	31.4	30.5	29.5
2	36.2	30.1	29.4	29.0
4	33.1	29.5	28.3	28.4
8	30.7	28.8	27.3	27.2
16	**29.5**	**28.6**	**27.3**	
32	29.9	29.3	27.6	

# mixture component	Phone Error Rate (%)			
	Maximum likelihood	Online w/o margin	Online w/ margin	Batch w/ margin
1	41.5	33.6	32.8	31.2
2	38.0	32.3	31.4	30.8
4	34.9	31.4	30.3	29.8
8	32.3	30.1	28.6	28.2
16	**30.8**	**29.7**	**28.8**	
32	31.8	30.9	29.0	

error rates; this discrepancy reflects the fact that the margin-based updates more closely track the Hamming distance (not the edit distance) between target and Viterbi phone sequences. Nevertheless, comparing to the results in Table II, we see that the gains from margin-based decoding exceed the gains from all other methods (batch and online) that do not incorporate large margin constraints.

For reference, table VII also reproduces previous results obtained from batch implementations of large margin training [12]. The objective function for batch training differed slightly from the ones we use in eq. (26) for online learning; specifically, the batch optimization minimized a soft-max approximation to the first term in eq. (26) using a projected subgradient method. The online updates do not quite match the performance of the batch implementation; however, they are simpler to implement and require fewer passes through the set of training utterances. Moreover, we will see in section V that the online updates can be extended in simple ways to achieve even further gains.

While Table VII quantifies the effects of margin-based decoding on error rates, Fig. 3 graphically illustrates the profound influence it exerts during training. To create this figure, we computed the Hamming distance between the Viterbi decoding $\mathbf{s}^*$ in eq. (3) and the margin-based decoding $\tilde{\mathbf{s}}^*$ in eq. (25) for each utterance during one online pass through the training corpus. The figure shows a histogram of these Hamming distances after they have been normalized by the number of frames in the utterance. The histogram's peak away from zero shows that margin-based decoding yields very different competing transcriptions for discriminative training than standard Viterbi decoding.

The frame and phone error rates from large margin training depend on the value of the margin scaling factor ρ. Fig. 4 shows this dependence for HMMs with 4-component GMMs in each state. More generally, for phone error rates on the development set, the optimal values of ρ were respectively 0.8, 1.0, 0.7, and 1.0 for HMMs with 1, 2, 4, and 8-component GMMs. Training with $\rho = 0$ (i.e., without margin-based decoding) produces the results shown in the middle columns of Table VII.

TABLE VIII
PHONE ERROR RATES FROM LARGE MARGIN TRAINING USING MANUALLY ALIGNED PHONETIC TRANSCRIPTIONS VERSUS FORCED ALIGNMENTS; SEE TEXT FOR DETAILS.

# mix	Phone Error Rate (%)	
	manual	forced
1	32.8	32.9
2	31.4	30.9
4	30.3	29.8
8	28.6	28.9
16	**28.8**	**28.2**
32	29.0	30.6

Fig. 5 illustrates the relatively fast convergence of online learning. The figure shows the frame error rates on the development data set during training. For all model sizes, most of the improvement from online learning occurs during the first 10-20 passes through the training corpus. Many more passes are typically required for convergence of batch methods.

Finally, we consider the applicability of our approach to other common training scenarios. While the TIMIT speech corpus has manually aligned phonetic transcriptions, most speech corpora do not have such information. For large margin training, our framework requires target state sequences that specify the hidden state in each frame of speech. When these alignments are not available from the corpus itself, what can we use in their place? The simplest option is to compute forced alignments of the training speech from whatever word or phonetic transcriptions are provided. To evaluate this option, we experimented with large margin training where for target state sequences, we used forced alignments generated by seed CD-HMMs trained by ML estimation. Table VIII compares the phone error rates from large margin training using manual versus forced alignments for different model sizes. Surprisingly, the results from forced alignments are comparable to (and sometimes even slightly better than) those obtained from the "ground truth" transcriptions. Thus it does not seem that precise knowledge of phoneme boundaries is required for large margin training, provided that forced alignments of reasonable quality can be generated from a seed model.

Our final experiment was motivated by the fact that many researchers choose not to use full covariance matrices in CD-

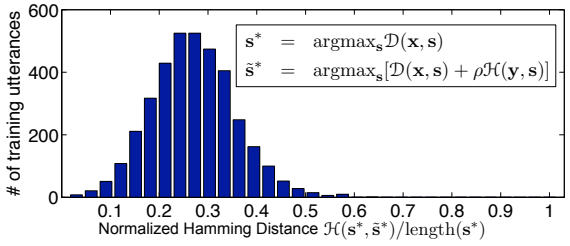

Fig. 3. Histogram of normalized Hamming distances between sequences from Viterbi and margin-based decoding. The distances were computed during the fifth iteration through the training corpus for the best-performing large margin HMM with sixteen Gaussian mixture components per hidden state.

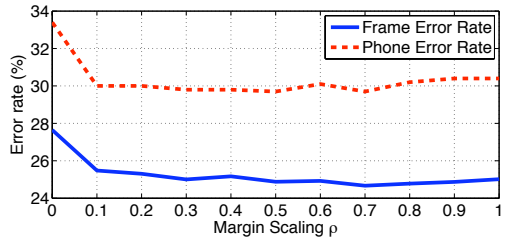

Fig. 4. Frame and phone error rates on the development set as a function of the margin scaling factor ρ. Results are shown for acoustic models with four Gaussian mixture components per hidden state.

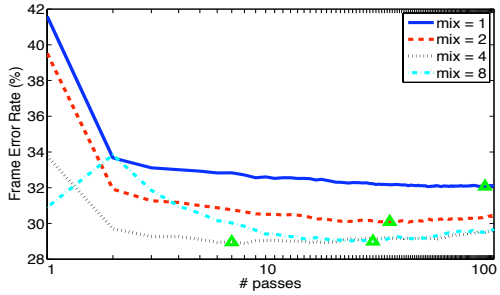

Fig. 5. Frame error rates on the development set during training. The triangles mark the best models obtained for different numbers of Gaussian mixture components.

HMMs for ASR. Table IX compares the results when HMMs with diagonal covariance matrices were estimated by ML versus online updates for large margin training. For the latter, we used slight variants of the online updates in eqs. (13)-(15); in particular, we constrained the covariance matrices to be diagonal, and we computed gradients with respect to the large-margin loss function in eq. (26). The results show that for purely diagonal covariance matrices, large margin training also yields lower frame and phone error rates than ML estimation. However, the improvements are not as substantial as those obtained from full covariance matrices using the parameterization in eq. (17).

TABLE IX
FRAME AND PHONE ERROR RATES FOR HMMS WITH DIAGONAL COVARIANCE MATRICES, AS OBTAINED BY MAXIMUM LIKELIHOOD (ML) ESTIMATION AND ONLINE UPDATES FOR LARGE MARGIN TRAINING. THE LEFT COLUMN SHOWS THE NUMBER OF MIXTURE COMPONENTS PER GMM.

#	Frame Error Rate (%)		Phone Error Rate (%)	
mix	ML	online	ML	online
1	44.0	39.2	46.8	43.5
2	40.0	35.6	43.3	39.3
4	37.6	34.0	41.1	37.1
8	34.8	32.3	37.5	35.3
16	34.1	31.0	36.4	34.1
32	32.5	31.0	34.5	33.0
64	31.5	31.4	33.5	33.5
128	30.6	**30.2**	32.8	**32.0**
256	31.6	31.4	33.6	33.4

V. ACOUSTIC FEATURE ADAPTATION

Acoustic features for ASR are typically computed by a front end which extracts them from short, sliding windows of speech. Most front ends compute mel-frequency cepstral coefficients (MFCCs) and higher-order derivatives of MFCCs that capture changes over time [43]. While the parameters of CD-HMMs are estimated from large amounts of speech, it is less common for the parameters in the front end to be systematically optimized in the same way.

Many researchers have noted this discrepancy and pursued approaches that blur the distinction between front and back ends in ASR. In particular, adaptive methods are increasingly being applied at all stages of pattern recognition, from the lowest levels of feature extraction to the highest levels of decision-making. Early influential work along these lines involved data-driven methods for robust feature extraction [44] and filterbank design [45], [46], [47]. More recent methods include: (i) heteroscedastic linear discriminant analysis (HLDA) [48] and neighborhood component analysis [49] to learn informative low dimensional projections of high dimensional acoustic feature vectors; (ii) stochastic gradient and second-order methods to tune parameters related to frequency warping and mel-scale filterbanks [50], [51]; (iii) maximum likelihood methods for speaker and environment adaptation [52], [53] that perform linear transformations of the acoustic feature space at test time; and (iv) extensions of popular frameworks for discriminative training, such as minimum phone error [54] and maximum mutual information [55], to learn accuracy-improving transformations and projections of the acoustic feature space.

In this section, we show how to extend the large margin updates in eq. (27) to learn a linear transformation of the acoustic feature space. The linear transformation is parameterized by a projection matrix which maps the cepstral coefficients from multiple adjacent analysis windows into a lower-dimensional acoustic feature vector. We derive online updates to adapt the elements of this projection matrix after the decoding of each training utterance.

The projection matrix affects how acoustic feature vectors are computed in every frame of speech. For this reason, small changes to the projection matrix can have large effects on recognition. This sensitivity presents a challenge for online learning, where the acoustic feature space is constantly adapted based on the statistics of individual training utterances. To mitigate the strongly biased gradients from individual utterances, we experimented with different schemes for regularization and parameter-tying. Our results show that parameter-tying helps to stabilize online learning by accumulating and averaging gradients across otherwise independent computations.

Our work is distinguished from previous schemes for feature adaptation in three ways. First, we consider how to jointly optimize the parameters in the front end along with the acoustic models in the back end. Second, the feature adaptation is driven by an objective function for large margin training, which seeks to separate the log-likelihoods of correct and incorrect transcriptions by an amount proportional to their Hamming distance. Third, we explore parameter-tying not across different mixture components or hidden states in the same HMM, but across different recognizers; in particular, we train several different recognizers in parallel while tying the feature projection matrices in their front ends.

A. Derivative features and linear projections

In most systems for ASR, the front end computes acoustic feature vectors from mel-frequency cepstral coefficients (MFCCs). Typically, the first $d_0 = 13$ MFCCs are used in this analysis. Due to co-articulation and other temporal effects, the MFCCs from one analysis window may contain information about the phonetic content in neighboring windows. To capture this information, most front ends also incorporate MFCCs from neighboring windows into their acoustic feature vectors. In particular, they compute derivative features, such as delta and delta-delta MFCCs, and augment the feature vector to include them.

The derivative features are computed by linearly combining MFCCs from neighboring analysis windows. The weights used to combine adjacent MFCCs are fixed and determined heuristically. Unlike most other parameters in modern speech recognizers, these weights in the front end are not typically adapted to optimize performance.

In this section, we consider how to optimize the linear transformation used to compute derivative features in conjunction with the back end for large margin HMMs (described in section IV). The standard derivative features are computed from a linear transformation of the raw MFCCs in nearby frames. Let u_t denote the $d_0 = 13$ MFCCs computed at time t, and let $\mathbf{v_t}$ denote the "stacked" MFCCs obtained by concatenating $4K+1$ consecutive frames $u_{t-2K}, u_{t-2K+1}, ..., u_t, ..., u_{t+2K}$ for some small value of K. Finally, let x_t denote the acoustic feature vector derived from the MFCCs at time t and their first and second-order derivatives. Then x_t and v_t are related by the linear transformation:

$$x_t = H_0 v_t, \tag{28}$$

where H_0 is the projection matrix whose entries approximate derivatives by finite differencing operations on nearby frames. Note that eq. (28) describes how acoustic feature vectors were computed for all the experiments described in previous sections of this paper.

The matrix H_0 is only one of many possible projection matrices that can be used to compute acoustic feature vectors from MFCCs in adjacent frames of speech. In this section, we explore different feature adaptation strategies for ASR. In particular, we consider how to learn more general projection matrices in the context of large margin training for HMMs.

B. Loss function for feature adaptation

Our approach builds on the online updates for large margin training in eq. (27). Let $\hat{x}$ denote a stacked feature vector of MFCCs from $4K+1$ adjacent windows, as described in section V-A, and let z denote the lower dimensional acoustic feature vector that appears in eq. (18). We seek a projection matrix $H \in \Re^{D \times d}$ that maps the high-dimensional vector $\hat{x}$ of stacked MFCCs to the low-dimensional acoustic feature vector z; then for each window, we can compute:

$$z = H\hat{x}, \quad \text{where} \quad \hat{x} = \begin{bmatrix} v \\ 1 \end{bmatrix}. \tag{29}$$

Note that H has one extra row and column than the projection matrix H_0 in eq. (28) due to the augmented feature vector z that appears in eq. (18) for large margin HMMs. In particular, we have $d = 3d_0 + 1$ and $D = (4K+1)d_0 + 1$, where $d_0 = 13$ is the number of MFCCs computed per window.

For acoustic feature adaptation in large margin HMMs, we update the projection matrix H and the parameter matrices Φ_{sc} so that the constraints in eq. (24) are satisfied for as many training utterances as possible. Let $\{(\hat{\mathbf{x}}_n, \mathbf{y}_n)\}_{n=1}^N$ denote the N labeled feature-state sequences in the training corpus, where the observations live in the high-dimensional feature space (before projection). For online learning, we examine one utterance at a time and compute the hidden state sequence by eq. (25). Analogous to sections III and IV, we define the loss function as:

$$\mathcal{L}(H, \Phi) = \sum_n \left[\max_{\mathbf{s} \neq \mathbf{y}_n} \left[\mathcal{D}(\hat{\mathbf{x}}_n, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}_n) \right] - \mathcal{D}(\hat{\mathbf{x}}_n, \mathbf{y}_n) \right]^+. \tag{30}$$

Eq. (30) differs from eq. (26) in two respects: first, the observed feature vectors in this context live in a much higher-dimensional space; second, in addition to the model parameters Φ, the loss function also depends on the feature projection matrix H.

The margin-based loss function in eq. (30) depends on the matrices Φ_{sc} and H through eqs. (8, 18) and (29). Specifically, we can write the discriminant function as:

$$\begin{aligned} \mathcal{D}(\hat{\mathbf{x}}, \mathbf{s}) &= \log \mathcal{P}(s_1) + \sum_{t=1}^{T-1} \log \mathcal{P}(s_{t+1} | s_t) \\ &\quad + \sum_{t=1}^{T} \log \sum_c e^{-\frac{1}{2} \hat{x}_t^\top H^\top \Phi_{sc} H \hat{x}_t}. \end{aligned} \tag{31}$$

Note that while eq. (31) depends on the high dimensional (stacked) cepstral feature vectors $\hat{x}_t \in \Re^D$, the loss function can be computed entirely in terms of the low dimensional features $z_t = H\hat{x}_t$. In fact, we can view $H^\top \Phi_{sc} H$ as storing a low-rank factorization of an inverse covariance matrix in the high dimensional space of unprojected cepstral features.

C. Parameter-tying

The loss function in eq. (30) can be minimized by alternately updating H and Φ_{sc}. However, we have noticed that small changes in the projection matrix H can drastically change the decoding results. This sensitivity is to be expected since the projection matrix H is used to calculate acoustic features in every frame of speech.

One way to reduce this sensitivity is to perform some sort of averaging. Batch training reduces this sensitivity by averaging over all the utterances in the training set. However, batch training does not exploit the fact that many training utterances convey redundant information. Some of the advantages of batch training can be obtained by online updates that average gradients over "mini-batches" of training utterances. For acoustic feature adaptation, however, we found that additional measures were needed.

The rest of this section describes a parameter-tying scheme that helps to mitigate the strongly biased gradients from individual training utterances. In this scheme, we tie the projection matrix H across several different recognizers whose

parameters are jointly updated after decoding each training utterance. By averaging the gradients across multiple recognizers, we hope to obtain more stable online updates.

Parameter-tying in CD-HMMs has been widely adopted for ASR [27], [28]. It has two main benefits: first, it reduces the memory footprint of speech recognizers, and second, it reduces the number of free parameters that must be estimated from limited training data. Our scheme for parameter-tying is subtly different than previous approaches. Typically, parameters are tied across different hidden states or mixture components in the same recognizer. In our scheme, however, we tie parameters across multiple different recognizers that are trained in parallel. These recognizers may have different model sizes (i.e., different numbers of hidden states and/or mixture components). By tying the projection matrix, however, we force all the recognizers to use the same front end.

Our approach is based on a global cost function for parallel training of multiple models or recognizers. We index each available model by the superscript α; thus, each model has its own (back-end) parameters Φ^α, as well as a shared (front-end) feature projection matrix H. The global cost function is given by:

$$\mathcal{L} \;=\; \sum_{\alpha,n} \left[\max_{\mathbf{s} \neq \mathbf{y}_n} \left[\mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{s}) + \rho \mathcal{H}(\mathbf{s}, \mathbf{y}_n) \right] - \mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{y}_n) \right]^+, \tag{32}$$

where $\mathcal{D}^\alpha(\hat{\mathbf{x}}, \mathbf{s})$ is the discriminant function for the model with parameters Φ^α. In our implementation, the available models are large margin HMMs with one hidden state per phone but different numbers of Gaussian mixture components per hidden state. Eq. (32) differs from eq. (30) only in the accumulation of information across models. Thus, the parameter-tying only affects the gradients for optimizing the tied projection matrix H, but not the gradients for optimizing each model's individual (non-tied) parameter matrix Φ^α.

D. Online updates

The objective function in eq. (32) lends itself to an alternating minimization procedure. Such a procedure alternates between two phases, one optimizing Φ while holding H fixed; the other optimizing H while holding Φ fixed. Because the optimization is susceptible to local minima, we must also consider carefully how to initialize the projection and parameter matrices in this context.

The online updates for minimizing eq. (32) are a straightforward extension of those in the previous section. We alternatingly update the projection and parameter matrices in the following way. First, we choose an utterance $(\hat{\mathbf{x}}_n, \mathbf{y}_n)$ at random from the training corpus. Then, for each individual model, we update its parameter matrix by:

$$\Phi^\alpha \;\leftarrow\; \Phi^\alpha \;+\; \eta_\Phi \frac{\partial}{\partial \Phi^\alpha} \left[\mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{y}_n) - \mathcal{D}^\alpha(\hat{\mathbf{x}}_n, \mathbf{s}_n^{\alpha *}) \right], \tag{33}$$

where the state sequence $\mathbf{s}_n^{\alpha *}$ is computed from the margin-based decoding in eq. (25). The right hand side of eq. (33) depends on the current value of the parameter matrix $\Phi_{\mathcal{M}}$ and the projection matrix H. We optionally repeat this online update for several additional utterances. Following these updates for the model parameter matrices, we perform updates for the feature projection matrix. Given an utterance $(\hat{\mathbf{x}}_m, \mathbf{y}_m)$ at random from the training corpus, we update the projection matrix H by:

$$H \;\leftarrow\; H \;+\; \eta_H \frac{\partial}{\partial H} \sum_\alpha \left[\mathcal{D}^\alpha(\hat{\mathbf{x}}_m, \mathbf{y}_m) - \mathcal{D}^\alpha(\hat{\mathbf{x}}_m, \mathbf{s}_m^{\alpha *}) \right]. \tag{34}$$

The right hand side of eq. (34) depends on the current value of the projection matrix H and the parameter matrices Φ. Note that unlike the update in eq. (33), all models contribute to the optimization of the projection matrix H through the summation in the gradient. For more stable learning, we often perform the update in eq. (34) in a mini-batch fashion, averaging over small sets of training utterances. We continue this procedure, alternately updating the GMM and projection matrix parameters whenever the results from margin-based decoding do not match the target transcriptions. The scalar learning rates η_Φ and η_H determine the step sizes; in practice, we tune them independently to achieve the fastest convergence.

E. Experiments

Our experiments had two main goals: first, to test whether feature adaptation can improve phoneme recognition beyond the usual gains of discriminative training; second, to investigate the potential benefits of parameter-tying in this context. Our baseline systems were discriminatively trained HMMs with traditional cepstra, delta-cepstra, and delta-delta-cepstra as features (Table VII). Our front end computed $d_0 = 13$ mel-frequency cepstral coefficients (MFCCs) in each analysis window; initial acoustic features were computed by linearly combining the cepstra across 13 consecutive analysis windows (i.e., including six windows on each side of the current window); see eq. (28). To adapt the acoustic feature space, we concatenate all 169 cepstral features from these 13 windows, append a constant scalar feature of value one, and then estimate a 40x170 projection matrix, as in eq. (29). We experimented on acoustic models of different sizes, with 1, 2, 4, 8, 16 or 32 Gaussian mixture components per hidden state.

Since the optimization for acoustic feature adaptation is highly nonlinear, the results can be sensitive to how model parameters are initialized. We used the following scheme to obtain the positive results in this paper. First, we initialized all discriminatively trained models by their maximum likelihood counterparts. Second, we initialized all models with feature adaptation by setting the upper left block of H equal to H_0; thus, the MFCCs from different windows were initially combined by computing standard *delta* and *delta-delta* features. Third, in some experiments, we constrained the initially zero elements of the projection matrix H to remain zero; in other words, though the features were reweighted, the sparsity pattern of the projection matrix was not allowed to change during learning. This constraint led to more reliable convergence in the models without parameter-tying.

Table X compares the frame and phone error rates of acoustic models trained in different ways: by maximum likelihood (ML) estimation, by large margin (LM) training (Section IV),

by large margin training with feature adaptation (LM+FA) but no parameter-tying, and by large margin training with feature adaptation and parameter-tying across models of different sizes (LM+FA+PT), using both sparse and full projection matrices H. All discriminatively trained models were initialized from the same ML baseline, thus starting from exactly the same performance. Also, for all these experiments, we fixed the margin-scaling parameter in eq. (30) as $\rho = 1$, rather than optimizing it on held-out data (which is somewhat expensive). This choice was based on the experiments in the previous section, where our results depended weakly on ρ as long as its value was greater than some small threshold.

The results in Table X show three general trends: first, that feature adaptation (LM+FA) improves performance beyond the already significant gains from large margin training (LM); second, that feature adaptation works best in conjunction with parameter-tying (LM+FA+PT) across different models; third, that the most general scheme for feature adaptation (without sparsity constraints on H) leads to the most improvement, provided that the learning is regularized in other ways. In particular, to obtain the results in the last column of Table X, we not only tied the full matrix H across different models; we also employed a parameter-averaging update for the full matrix H, as described in eq. (12). Without both parameter-tying across models and parameter-averaging over time, learning with full matrices H yielded worse results on both the development and test sets.

The exceptions to these trends are also revealing. For example, in the largest model with 8 Gaussian mixture components per hidden state, the frame and phone error rates are not improved by feature adaptation without parameter-tying; in fact, they are slightly worse. The worse performance may be due to overfitting and/or unreliable convergence. However, the performance in this model is improved when the feature adaptation in the front end is tied across different recognizers. The parameter-tying appears to mitigate the challenges of feature adaptation in large models. Specifically, it appears to dampen the fluctuations that arise in online learning, when updates are based on the decoding of individual training utterances. By tying the projection matrix across different model sizes, the larger model benefits from information that is accumulated across different recognizers. Moreover, without parameter-tying, we observed that much smaller learning rates were required to obtain small but consistent improvements from acoustic feature adaptation.

Finally, we comment on convergence issues. In general, parameter-tying led to better results but not necessarily faster training: that is, roughly the same number of passes through the training data were required to converge (as measured by performance on the validation set). However, while the update rule in eq. (34) accumulates information across different models, we can always distribute the computation across multiple nodes, summing up the gradients from individual models as necessary. When implemented in this way, the total running time for learning is essentially equal to the individual running time of the largest model in the ensemble of recognizers. This approach exploits parallelism in a similar way as many implementations of batch training for large-scale ASR.

TABLE X
FRAME AND PHONE ERROR RATES ON THE TIMIT TEST SET FOR ACOUSTIC MODELS OF VARYING SIZE, AS OBTAINED BY MAXIMUM LIKELIHOOD (ML) ESTIMATION AND ONLINE UPDATES FOR LARGE MARGIN TRAINING (LM), FEATURE ADAPTATION (FA), AND PARAMETER-TYING (PT). SEE TEXT FOR DETAILS. THE BEST RESULTS IN EACH ROW ARE SHOWN IN BOLD.

# of mix	Frame Error Rate (%)				
	H_0		sparse H		full H
	ML	LM	LM+FA	LM+FA+PT	LM+FA+PT
1	39.7	30.5	30.4	29.2	29.2
2	36.2	29.4	28.1	28.1	27.8
4	33.1	28.3	27.4	27.4	27.5
8	30.7	27.3	**27.4**	**26.6**	**26.4**
16	**29.5**	**27.3**	28.2	27.5	27.5
32	29.9	27.6	29.7	28.1	28.5

# of mix	Phone Error Rate (%)				
	H_0		sparse H		full H
	ML	LM	LM+FA	LM+FA+PT	LM+FA+PT
1	41.5	32.8	32.2	31.9	31.5
2	38.0	31.4	29.6	30.3	29.5
4	34.9	30.3	29.3	29.2	29.1
8	32.3	28.6	**28.8**	**27.8**	**27.7**
16	**30.8**	**28.8**	29.6	28.6	28.5
32	31.8	29.0	31.3	29.6	30.0

VI. DISCUSSION

In this paper, we have explored the potential of online updates for discriminative training of HMMs. We have also shown that such updates can be used for large margin training and acoustic feature adaptation, yielding further improvements in performance. We conclude by summarizing our main contributions.

In section III, we identified two reparameterizations of GMMs that lead to more effective online updates. The reparameterization in eq. (17) aggregates the mean and covariance matrix parameters of GMMs, yielding simpler gradients and eliminating the need for multiple different learning rates. The further reparameterization in eq. (20) leads to significantly faster convergence. Based on extensive experimental results, we identified these reparameterizations, as well as the averaging in eq. (12) as best practices for discriminative training of HMMs using online updates. Moreover, as shown in Table II, acoustic models trained in this way performed better than acoustic models trained by popular batch approaches such as CML and MCE.

In section IV, we showed how to extend these online updates to incorporate large margin constraints. Large margin training led to further improvements in frame and phone error rates. Though the online updates for large margin training did not match the performance of previous batch implementations, they were simpler to implement and required fewer passes through the set of training utterances.

Finally, in section V, we used online updates for end-to-end training of speech recognizers. In particular, we experimented with alternating updates that jointly adapted the features computed by the front end in conjunction with large margin training of HMMs. For our best results, we developed a novel parameter-tying scheme that tied the feature projection matrices of multiple different recognizers trained in parallel. This scheme yielded further improvements in frame and phone error

rates across all model sizes; the results also surpassed those from previous batch implementations of large margin training (without acoustic feature adaptation). Though acoustic features could in principle be adapted within a batch framework, the required implementation and experimentation would be much more unwieldy.

One interesting direction for future work is to investigate other distance penalties besides the Hamming distance in eq. (24). Perhaps better results could be obtained by weighted distances that incorporate prior knowledge about likely phonetic confusions [56]. Also, it would be interesting to consider more sophisticated distances (such as edit distances) that only penalize word transcription errors. However, such distances are not as straightforward to incorporate into our framework.

Another interesting direction for future work is to explore different and larger sets of acoustic features. For example, the mel-frequency cepstra are themselves computed from a linear projection of the log-magnitude spectra. We could use these log-magnitude spectra as the high dimensional features instead of the cepstra and still initialize our models with the baseline performance of existing approaches. The projection matrices in this case would not only be significantly larger, but more or less completely dense. Presumably, larger training corpora would be required to estimate discriminative projection matrices in this case. However, this regime is precisely where one expects the biggest pay-off from online methods, as we have considered in this paper.

ACKNOWLEDGMENTS

We thank the reviewers for extensive comments and suggestions for improvement on all parts of the paper. Fei Sha would like to acknowledge useful discussions with Samy Bengio, Li Deng, and Brian Kingsbury. This work was supported in part by NSF Award 0812576. Fei Sha is also partially supported by the Charles Lee Powell Foundation.

REFERENCES

[1] X. Huang, A. Acero, and H.-W. Hon, *Spoken language processing*. Prentice-Hall, 2001.

[2] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-86)*, Tokyo, 1986, pp. 49–52.

[3] A. Nádas, "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*, vol. 31, no. 4, pp. 814–817, 1983.

[4] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Sig. Proceedings*, vol. 40, no. 12, pp. 3043–3054, 1992.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[6] V. N. Vapnik, *Statistical Learning Theory*. New York, NY: John Wiley & Sons, 1998.

[7] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *International Conference on Machine Learning (ICML-04)*, Banff, Canada, 2004, pp. 104–112.

[8] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[9] H. Jiang, X. Li, and C. Liu, "Large margin hidden markov models for speech recognition," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14, no. 5, pp. 1584–1595, 2006.

[10] J. Li, M. Yuan, and C. Lee, "Approximate test risk bound minimization through soft margin estimation," *IEEE Trans. on Speech, Audio and Language Processing*, vol. 15, no. 8, pp. 2392–2404, 2007.

[11] D. Yu, L. Deng, X. He, and A. Acero, "Large-margin minimum classification error training for large-scale speech recognition tasks," in *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-07)*, vol. 4, 2007, pp. 1137–1140.

[12] F. Sha and L. K. Saul, "Large margin training of continuous density hidden markov models," in *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, J. Keshet and S. Bengio, Eds. Wiley-Blackwell, 2009, pp. 101–114.

[13] P. C. Woodland and D. Povey, "Large scale discriminative training for speech recognition," in *Proceedings of Automatic Speech Recognition (ASR-2000)*, 2000, pp. 7–16.

[14] L. Bottou and Y. LeCun, "Large scale online learning," in *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press, 2004.

[15] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. MIT Press, 2008, vol. 20, pp. 161–168.

[16] L. F. Lamel, R. H. Kassel, and S. Seneff, "Speech database development: design and analysis of the acoustic-phonetic corpus," in *Proceedings of the DARPA Speech Recognition Workshop*, L. S. Baumann, Ed., 1986, pp. 100–109.

[17] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification," in *Proceedings of Nineth European Conference on Speech Communication and Technology (EuroSpeech 2005)*, Lisbon, 2005, pp. 1117–1120.

[18] S. Petrov, A. Pauls, and D. Klein, "Learning structured models for phone recognition," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 897–905.

[19] S. M. Siniscalchi, P. Schwarz, and C.-H. Lee, "High-accuracy phone recognition by combining high-performance lattice generation and knowledge based rescoring," in *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing (ICASSP-07)*, vol. 4, 2007, pp. 869–872.

[20] L. Deng, D. Yu, and A. Acero, "Structured speech modeling," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 14, no. 5, pp. 1492–1504, 2006.

[21] C. C. Cheng, F. Sha, and L. K. Saul, "Matrix updates for perceptron training of continuous density hidden markov models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009, pp. 153–160.

[22] F. Sha and L. K. Saul, "Large margin hidden Markov models for automatic speech recognition," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hofmann, Eds. Cambridge, MA: MIT Press, 2007, pp. 1249–1256.

[23] ——, "Comparison of large margin training to other discriminative methods for phonetic recognition by hidden Markov models," in *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-07)*, Honolulu, HI, 2007, pp. 313–316.

[24] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariahy, "Boosted MMI for model and feature-space discriminative training," in *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-08)*, 2008, pp. 4057–4060.

[25] G. Saon and D. Povey, "Penalty function maximization for large margin HMM training," in *Proceedings of Interspeech-2008*, 2008, pp. 920–923.

[26] C. C. Cheng, F. Sha, and L. K. Saul, "A fast online algorithm for large margin training of continuous density hidden markov models," in *Proceedings of Interspeech-2009*, 2009, pp. 668–671.

[27] S. J. Young, "The general use of tying in phoneme-based HMM speech recognisers," in *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-92)*, 1992, pp. 569–572.

[28] V. Digalakis and H. Murveit, "High-accuracy large-vocabulary speech recognition using mixture tying and consistency modeling," in *Proceedings of the workshop on Human Language Technology (HLT-94)*. Association for Computational Linguistics, 1994, pp. 313–318.

[29] C. C. Cheng, F. Sha, and L. K. Saul, "Acoustic feature adaptation in large margin hidden Markov models," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-09)*, Merano, Italy, 2009, pp. 87–92.

[30] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in *Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2002, vol. 14, pp. 841–848.

[31] P. Liang and M. I. Jordan, "An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators," in *International Conference on Machine Learning (ICML)*, 2008, pp. 584–591.

[32] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1641–1648, 1988.

[33] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[34] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms," in *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-02)*, vol. 10, 2002, pp. 1–8.

[35] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 173–180.

[36] T. Kudo and Y. Matsumoto, "Chunking with support vector machines," in *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*. Morristown, NJ, USA: Association for Computational Linguistics, 2001, pp. 1–8.

[37] F. Sha and L. K. Saul, "Large margin hidden markov models for automatic speech recognition," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007, pp. 1249–1256.

[38] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Artificial Intelligence, LNAI 3176, O. Bousquet and U. von Luxburg, Eds. Berlin: Springer Verlag, 2004, pp. 146–168.

[39] L. Bottou, Y. Le Cun, and Y. Bengio, "Global training of document processing systems using graph transformer networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-97)*, Puerto Rico, 1997, pp. 489–493.

[40] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," in *Machine Learning*, 1999, pp. 277–296.

[41] C. Gentile, "A new approximate maximal margin classification algorithm," *J. Mach. Learn. Res.*, vol. 2, pp. 213–242, 2002.

[42] J. Keshet, S. Shalev-Shwartz, S. Bengio, Y. Singer, and D. Chazan, "Discriminative kernel-based phoneme sequence recognition," in *Proceedings of Interspeech-06*, 2006, pp. 1284–1287.

[43] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[44] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky, "Feature extraction using non-linear transformation for robust speech recognition on the aurora database," *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 2, pp. II1117–II1120, 2000.

[45] N. Malayath and H. Hermansky, "Data-driven spectral basis functions for automatic speech recognition," *Speech Communication*, vol. 40, no. 4, pp. 449–466, 2003.

[46] H. Sawai, A. Waibel, P. Haffner, M. Miyatake, and K. Shikano, "Parallelism, Hierarchy, Scaling in Time-Delay Neural Networks for Spotting Japanese Phonemes/CV-Syllables," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2. Washington D.C.: IEEE, New York, 1989, pp. 81–88.

[47] P. Haffner and A. Waibel, "Multi-state time delay neural networks for continuous speech recognition," in *Advances in Neural Information Processing Systems 4*, 1991, pp. 135–142.

[48] M. K. Omar and M. Hawegawa-johnson, "Maximum conditional mutual information projection for speech recognition," in *Proceedings of Eurospeech 2003*, 2003, pp. 505–509.

[49] N. Singh-Miller, M. Collins, and T. J. Hazen, "Dimensionality reduction for speech recognition using neighborhood components analysis," in *Proceedings of Interspeech-07*, 2007, pp. 1158–1161.

[50] K. Visweswariah and R. Gopinath, "Adaptation of front end parameters in a speech recognizer," in *Proceedings of the International Conference on Spoken Language Processing*, 2004, pp. 21–24.

[51] S. Balakrishnan, K. Visweswariah, and V. Goe, "Stochastic gradient adaptation of front-end parameters," in *Proceedings of Interspeech-2004*, 2004, pp. 1–4.

[52] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[53] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR transforms as features in speaker recognition," in *Proceedings of INTERSPEECH-2005*, 2005, pp. 2425–2428.

[54] B. Zhang and S. Matsoukas, "Minimum phoneme error based heteroscedastic linear discriminant analysis for speech recognition," in *Proceedings of the International Conference of Acoustic, Speech and Signal Processing (ICASSP-05)*, 2005, pp. 925–928.

[55] J. McDonough, M. Wlfelc, and E. Stoimenov, "On maximum mutual information speaker-adapted training," *Computer Speech and Language*, vol. 22, no. 2, pp. 130–147, 2008.

[56] O. Dekel, J. Keshet, and Y. Singer, "An online algorithm for hierarchical phoneme classification," in *Machine Learning for Multimodal Interaction*, 2005, pp. 146–158.