Support Vector Machine Training for Improved Hidden Markov Modeling

Alba Sloin and David Burshtein, *Senior Member, IEEE*

***Abstract*—We present a discriminative training algorithm, that uses support vector machines (SVMs), to improve the classification of discrete and continuous output probability hidden Markov models (HMMs). The algorithm uses a set of maximum-likelihood (ML) trained HMM models as a baseline system, and an SVM training scheme to rescore the results of the baseline HMMs. It turns out that the rescoring model can be represented as an unnormalized HMM. We describe two algorithms for training the unnormalized HMM models for both the discrete and continuous cases. One of the algorithms results in a single set of unnormalized HMMs that can be used in the standard recognition procedure (the Viterbi recognizer), as if they were plain HMMs. We use a toy problem and an isolated noisy digit recognition task to compare our new method to standard ML training. Our experiments show that SVM rescoring of hidden Markov models typically reduces the error rate significantly compared to standard ML training.**

***Index Terms*—Discriminative training, hidden Markov model (HMM), speech recognition, support vector machine (SVM).**

I. Introduction

THE HIDDEN Markov model (HMM) plays an important role in a variety of applications, including speech modeling and recognition and protein sequence analysis. Typically one assigns an HMM to each class, and estimates its parameters from some training database using the maximum likelihood (ML) approach. The recognition of an observed sequence that represents some unknown class can then proceed using the estimated HMM parameters. Although the ML approach is asymptotically unbiased and achieves the Cramer-Rao lower bound, it is not necessarily the optimal approach in terms of minimum classification error. If the assumed model is incorrect or the training set is not large enough, the optimal properties of ML training do not hold. In such cases, it is possible to benefit, in terms of lower error rates, from discriminative training methods, that consider all the training examples in the training set and train all the models simultaneously.

Manuscript received September 3, 2006; revised May 11, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ilya Pollak. This work was presented in part at the 24th IEEE Conference of Electrical And Electronics Engineers, Eilat, Israel, November 15–17, 2006. This work was supported in part by the KITE Consortium of the Israeli Ministry of Industry and Trade, by Muscle, a European network of excellence funded by the EC 6th framework IST programme, and by a fellowship from The Yitzhak and Chaya Weinstein Research Institute for Signal Processing at Tel-Aviv University.

The authors are with the School of Electrical Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel (e-mail: alba@eng.tau.ac.il; burstyn@eng.tau.ac.il).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2007.906741

One of the powerful tools for pattern recognition that uses a discriminative approach is the support vector machine (SVM). SVMs use linear and non-linear separating hyperplanes for data classification. However, since SVMs can only classify fixed length data vectors, this method can not be readily applied to tasks involving variable length data classification. The variable length data has to be transformed into fixed length vectors before SVMs can be used.

Several attempts have been made to incorporate the SVM method into variable length data classification systems. The SVM-Fisher method [1] offers a way of combining generative models like HMMs with discriminative methods like SVMs. Smith and Gales [2] applied the Fisher kernel to the speech recognition problem and provided insight in support of the Fisher kernel approach. In [3], the SVM-Fisher method was extended and applied to the problem of speaker verification using Gaussian mixture models (GMMs). In [4] the Gaussian DTW kernel (GDTW) was introduced. GDTW is based on the dynamic time warping (DTW) technique for pattern recognition and on the Gaussian kernel. In [5], a discriminative algorithm for phoneme alignment that uses an SVM-like approach is presented. In [6] a hybrid SVM/HMM system is presented. A set of baseline HMMs is used to segment the training data and transform it into fixed length vectors, and a set of SVM models are used for rescoring.

In this paper, we present a new algorithm that uses a set of ML trained HMM models as a baseline system, and an SVM training scheme to rescore the results of the baseline HMMs. In [7] we first presented our method for discrete HMMs. In this paper we discuss both discrete and continuous HMMs. In Section II, we give a short overview of SVMs and SVM training techniques. In Section III, we describe our algorithm for the discrete HMM case, and two methods for training the SVM models. In Section IV, we do the same for a continuous density HMM. In Section V, we assess the performance of our algorithms on a toy problem and on an isolated noisy digit recognition task. We compare the results of our two new training methods to the results achieved using standard ML training. Although our primary application in this work is automatic speech recognition, the same algorithms can be used in other applications that employ hidden Markov modeling.

II. Background on SVMs

The SVM [8]–[10], is a powerful machine learning tool that has been widely used in the field of pattern recognition.

Let $(\mathbf{x}_n, c_n)$, $n = 1, \ldots, N$, $c_n \in \{-1, 1\}$ be a set of vectors in $\mathbb{R}^d$ and their corresponding labels. We refer to this set as the

training set. Let $\phi(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^p$, where $p \geq d$, be some mapping from the vector space into some higher dimensional feature space. The support vector machine optimization problem attempts to obtain a good separating hyperplane between the two classes in the higher dimensional space. It is defined as follows:

$$\begin{aligned} &\min_{\xi,\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ &\quad \text{s.t.} \quad c_n(\langle \mathbf{w}, \phi(\mathbf{x}_n)\rangle + b) \geq 1 - \xi_n,\ n = 1,\ldots,N \\ &\qquad \xi_n \geq 0,\ n = 1,\ldots,N \end{aligned} \tag{1}$$

where $\langle \cdot,\cdot \rangle$ denotes an inner product between two vectors, and C is some constant that can be determined using a cross validation process. The Lagrangian dual problem of (1) is

$$\begin{aligned} &\max_{\alpha_n} \sum_n \alpha_n - \frac{1}{2}\sum_{n,j}\alpha_n\alpha_j c_n c_j K(\mathbf{x}_n,\mathbf{x}_j) \\ &\quad \text{s.t.} \quad 0 \leq \alpha_n \leq C,\ n = 1,\ldots,N \\ &\qquad \sum_{n=1}^{N}\alpha_n c_n = 0 \end{aligned} \tag{2}$$

where $K(\mathbf{x}_n,\mathbf{x}_j) \triangleq \langle \phi(\mathbf{x}_n), \phi(\mathbf{x}_j)\rangle$ is referred to as the kernel function. The choice of $K(\mathbf{x}_1,\mathbf{x}_2) = \langle \mathbf{x}_1,\mathbf{x}_2\rangle$ leads to a linear SVM. Since the optimization problem is convex and Slater's regularity conditions hold, the dual problem can be solved instead of the primal one, and both yield the same value (the minimum of the primal equals the maximum of the dual). The solution to this problem can be obtained using the efficient sequential minimal optimization (SMO) algorithm [11]. A new vector $\mathbf{x}$ will be classified as a member of the class with label 1 if

$$\langle \mathbf{w}, \phi(\mathbf{x})\rangle + b > 0$$

and as a member of the class with label -1 otherwise. The expression $\langle \mathbf{w}, \phi(\mathbf{x})\rangle + b$ can be shown to be equivalent to

$$\sum_n \alpha_n c_n K(\mathbf{x}_n,\mathbf{x}) + b$$

where α_n is the solution of the dual problem, (2). Since all the computations are done using the kernel function, there is no need to work in the higher dimensional space. The computation of the kernel function may be very simple, even if the underlying space is of very high or even infinite dimension.

The SVM algorithm described so far can only deal with the binary case, where there are only two classes. There are several possibilities of extending the binary class SVM into a multiclass SVM. We will describe two such possible extensions. The first is a natural extension referred to as the one against all method (see [12]). The second is a transformation to the one class problem [13].

A. The One Against All Method

The one against all algorithm solves the multi-class problem by training a binary SVM for each of the M classes. Each SVM is trained using all the data vectors from all classes. The data vectors that belong to the class are used as positive examples and all other vectors are used as negative examples. More formally, let $\{\mathbf{x}_n, c_n\}_{n=1}^N$ be a set of data vectors and their corresponding labels, where $\mathbf{x}_n \in \mathbb{R}^d$ and $c_n \in \{1,\ldots,M\}$. Let $SV_m = (\mathbf{w}_m, b_m), m = 1,\ldots,M$, be a set of M SVMs such that $\mathbf{w}_m \in \mathbb{R}^d$ and $b_m \in R$. Suppose for simplicity that $\phi(\mathbf{x}) = \mathbf{x}$. The mth model is trained using $\{\mathbf{x}_n, c_n{}'\}_{n=1}^N$, where

$$c_n' = \begin{cases} 1 & \text{if } c_n = m \\ -1 & \text{otherwise.} \end{cases}$$

A new data vector $\mathbf{x}$ will be classified as a member of class m if

$$\langle \mathbf{w}_m, \mathbf{x}\rangle + b_m > \langle \mathbf{w}_{m'}, \mathbf{x}\rangle + b_{m'} \quad \forall m' \neq m$$

B. The One Class Transformation Method

In the one class method [13], M binary SVM models are trained simultaneously using all the training data. Again, let $\{\mathbf{x}_n, c_n\}_{n=1}^N$ be a set of data vectors and their corresponding labels, where $\mathbf{x}_n \in \mathbb{R}^d$ and $c_n \in \{1,\ldots,M\}$. A reasonable multiclass SVM optimization criterion is

$$\begin{aligned} &\min_{\xi,\{\mathbf{w}_m,b_m\}_{m=1}^M} \frac{1}{2}\sum_{m=1}^{M}\|\mathbf{w}_m\|^2 + C\sum_{n,j,m}\xi_{n,j,m} \\ &\quad \text{s.t. } \langle \mathbf{w}_m,\mathbf{x}_n\rangle + b_m - (\langle \mathbf{w}_j,\mathbf{x}_n\rangle + b_j) \geq 1 - \xi_{n,j,m}, \\ &\quad n = 1,\ldots,N,\ m = c_n,\ j \in \{1,\ldots,M\} \setminus m \\ &\qquad \xi_{n,j,m} \geq 0, n = 1,\ldots,N, m = c_n, j \in \{1,\ldots,M\} \setminus m. \end{aligned}$$

This formulation aims to train M SVMs such that the score given to each data vector by the correct model is higher than that given to it by the rest of the models.

The solution to this problem has high complexity. It can, however, be slightly modified and transformed into a simpler one class problem by adding the b_m terms to the objective function

$$\begin{aligned} &\min_{\xi,\{\mathbf{w}_m,b_m\}_{m=1}^M} \frac{1}{2}\sum_{m=1}^{M}(\|\mathbf{w}_m\|^2 + b_m^2) + C\sum_{n,j,m}\xi_{n,j,m} \\ &\quad \text{s.t.} \quad \langle \mathbf{w}_m,\mathbf{x}_n\rangle + b_m - (\langle \mathbf{w}_j,\mathbf{x}_n\rangle + b_j) \geq 1 - \xi_{n,j,m}, \\ &\quad n = 1,\ldots,N, m = c_n, j \in \{1,\ldots,M\} \setminus m \\ &\qquad \xi_{n,j,m} \geq 0, n = 1,\ldots,N, m = c_n, j \in \{1,\ldots,M\} \setminus m. \end{aligned} \tag{3}$$

This modified problem was shown to give results that are very similar to that of the original one [14]. The modified problem can be reformulated as a one class SVM problem using the following notation: Let $\mathbf{w}$ denote the concatenation of the M SVM vector parameters $\mathbf{w} = (\mathbf{w}_1, b_1, \mathbf{w}_2, b_2, \ldots, \mathbf{w}_M, b_M)$, and let $\mathbf{z}_{n,j,m} = (0,\ldots,0,\mathbf{x}_n,1,\ldots,0,-\mathbf{x}_n,-1,0,\ldots,0)$ such that

$$\langle \mathbf{w}, \mathbf{z}_{n,j,m}\rangle = \langle \mathbf{w}_m,\mathbf{x}_n\rangle + b_m - \langle \mathbf{w}_j,\mathbf{x}_n\rangle - b_j.$$

Using this notation, we can rewrite (3) as

$$\begin{aligned} &\min_{\xi,\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n,j,m}\xi_{n,j,m} \\ &\quad \text{s.t.} \quad \langle \mathbf{w}, \mathbf{z}_{n,j,m}\rangle \\ &\qquad \geq 1 - \xi_{n,j,m}, n = 1,\ldots,N, m = c_n, j \in \{1,\ldots,M\} \setminus m \\ &\qquad \xi_{n,j,m} \geq 0, n = 1,\ldots,N, m = c_n, j \in \{1,\ldots,M\} \setminus m \end{aligned}$$

which is a one class SVM optimization problem that can be solved efficiently using a slightly modified SMO algorithm [11].

III. SVM Rescoring of Discrete HMMs

In this section, we focus on discrete HMMs, and propose a discriminative algorithm that uses a set of ML trained HMMs as a baseline system, and the SVM training scheme to rescore the results of the baseline system. We begin with the problem formulation, followed by the description of a variable to fixed length data transformation for discrete HMMs.

A. Problem Formulation

Let $\mathbf{o}=(o_1,o_2,\ldots,o_T)$ be some observed sequence, whose elements o_t take values in a finite set of symbols $\{1,\ldots,S\}$, i.e., $o_t\in\{1,\ldots,S\}$. Also consider an HMM over an alphabet of size S, with L states and with a parameter set denoted by θ. The parameter set θ is comprised of discrete output probability distributions $\zeta_i(\cdot)$, $i\in\{1,\ldots,L\}$ and transition probabilities $a_{i,j}$, $i,j\in\{1,\ldots,L\}$. The probability that the HMM assigns to the observation $\mathbf{o}$ and the state sequence $\tilde{\mathbf{q}}=(\tilde{q}_1,\ldots,\tilde{q}_T)$ is

$$P(\mathbf{o},\tilde{\mathbf{q}}\mid\theta)=\left(\prod_{t=1}^{T-1}\zeta_{\tilde{q}_t}(o_t)a_{\tilde{q}_t,\tilde{q}_{t+1}}\right)\cdot\zeta_{\tilde{q}_T}(o_T).$$

The probability that this HMM assigns to $\mathbf{o}$ is obtained by summing over all possible state sequences, $\tilde{\mathbf{q}}$

$$P(\mathbf{o}\mid\theta)=\sum_{\tilde{\mathbf{q}}}\left(\prod_{t=1}^{T-1}\zeta_{\tilde{q}_t}(o_t)a_{\tilde{q}_t,\tilde{q}_{t+1}}\right)\cdot\zeta_{\tilde{q}_T}(o_T).$$

We consider the following problem. Suppose that there are M different classes and M corresponding HMMs. The parameter set of the mth HMM is denoted by θ_m. Suppose that the prior probability of class m is p_m. Given the observed sequence, $\mathbf{o}$, we wish to predict its class. If the parameter vectors, θ_m, were known, then we could use the following maximum *a posteriori* (MAP) classifier that minimizes the classification error

$$\hat{m}=\arg\max_m\{p_mP(\mathbf{o}\mid\theta_m)\}. \tag{4}$$

In our problem, however, the parameter vectors, θ_m, are unknown. Thus before applying the MAP classifier, (4), we need to estimate them using a training database, $\mathcal{O}=(\mathbf{o}^1,\ldots,\mathbf{o}^N)$, $\mathcal{C}=(c^1,\ldots,c^N)$ where $c^n\in\mathcal{W}=\{1,\ldots,M\}$ is the true label of the time series observation, $\mathbf{o}^n$. The standard parameter estimation method uses the ML approach, according to which θ_m is selected so as to maximize

$$\sum_{n\,:\,c^n=m}\log p(\mathbf{o}^n\mid\theta_m) \tag{5}$$

which is the log-likelihood of the observations whose true class is m.

To implement this maximization, one typically applies the expectation-maximization (EM) method [15], that yields a local maximum of (5). A lower complexity alternative to (4) is the classification rule

$$\hat{m}=\arg\max_{m,\tilde{\mathbf{q}}}\{p_mP(\mathbf{o},\tilde{\mathbf{q}}\mid\theta_m)\}. \tag{6}$$

In our case, where the probabilistic model is an HMM, (4) is implemented by the forward algorithm while (6) is implemented by the Viterbi algorithm, and it is well known in the speech recognition literature (e.g., [16, Sec. 8.2.3, p. 388]) that both approaches yield similar results. Similarly, a lower complexity good alternative to maximizing (5) is to maximize

$$\sum_{n\,:\,c^n=m}\log\max_{\tilde{\mathbf{q}}}p(\mathbf{o}^n,\tilde{\mathbf{q}}\mid\theta_m). \tag{7}$$

Here we attempt to find the best parameter vector, θ_m, and state sequences, $\tilde{\mathbf{q}}$, for each observation n for which $c^n=m$. In our HMM case, (5) is implemented by the Baum-Welch (EM) algorithm, while (7) is implemented using the segmental K-means algorithm [17]–[19, Sec. 6.15.2, pp. 382–383] that applies a two steps iterative algorithm. In the first step, it obtains the best segmentation (state sequence) corresponding to each data sample n (for which $c^n=m$). In the second step, it re-estimates the parameter vector θ_m using these segmentations. Relations between the Baum Welch and Segmental K-means algorithms were studied in [20].

If our HMM parametric model is correct then it is well known that the ML estimator is asymptotically unbiased and efficient (i.e., it achieves the Cramer-Rao lower bound on estimation error). Thus, if the parametric model is accurate, and there is a sufficient amount of training data, then ML estimation (5) [or the alternative (7)] together with MAP classification (4) [or (6)] is a successful combination, even though it is not guaranteed to minimize the error rate even under these ideal conditions. In practice, however, these two assumptions may not hold. For example, in speech recognition, where HMM modeling is the standard approach, the true model is in fact unknown. Furthermore, ML training of θ_m considers only the observations $\mathbf{o}^n$ in the database whose true class is m. That is, ML training considers only positive examples, and neglects all the other observations in the training database, whose class is different than m. Discriminative training methods, on the other hand, attempt to train the parameters θ_m, such that for positive training examples (for which $c^n=m$) the MAP score $p_mP(\mathbf{o}\mid\theta_m)$ (or alternatively, $p_m\max_{\tilde{\mathbf{q}}}P(\mathbf{o},\tilde{\mathbf{q}}|\theta_m)$) would be high, and for negative training examples (for which $c^n\neq m$) the score would be lower. In the following, we show how such discriminative training can be realized using an SVM.

Note that our model is different than the conditional Markov random field considered, e.g., in [21]–[25] where, conditioned on the observation sequence, the label sequence is modeled by a Markov random field. These works usually assume a supervised or semisupervised training, where the label sequence is known at least for part of the training database. Recently, computationally intensive algorithms were also suggested for the more difficult case of unsupervised training of a conditional Markov random field model [24], [25].

B. A Variable to Fixed Length Data Transformation

Let $\mathbf{q}=(q_1,\ldots,q_T)$ denote the most likely state sequence corresponding to $\mathbf{o}$ according to some given HMM with parameter vector θ, i.e.

$$\mathbf{q}=\arg\max_{\tilde{\mathbf{q}}}P(\mathbf{o},\tilde{\mathbf{q}}\mid\theta).$$

We now describe a transformation that yields a new vector $\mathbf{T} = \mathbf{T}(\mathbf{o}, \mathbf{q})$ from $\mathbf{o}$ and $\mathbf{q}$. The vector $\mathbf{T}$, whose length is $LS+L^2+1$, is composed of the vectors $\boldsymbol{\phi}_i$, $i = 1, \ldots, L$, the vectors $\boldsymbol{\psi}_i$, $i = 1, \ldots, L$ and the scalar γ.

$$\mathbf{T} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_L, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_L, \gamma).$$

The vector $\boldsymbol{\phi}_i$ describes the count (nonnormalized empirical distribution) of the symbols that were emitted at state i, as determined by $\mathbf{q}$. For example $\boldsymbol{\phi}_i = (1, 0, 2, \ldots, 0)$ means symbol 1 was emitted once and symbol 3 was emitted twice at state i. More formally, let $\tau_i \triangleq \{t \mid q_t = i\}$, and let $\mathbf{e}_l^S$ denote an identity vector of length S whose lth element is 1 (e.g., $\mathbf{e}_1^S = (1, 0, 0, 0 \ldots, 0)$), then

$$\boldsymbol{\phi}_i = \sum_{t \in \tau_i} \mathbf{e}_{o_t}^S. \tag{8}$$

Similarly, the vector $\boldsymbol{\psi}_i$ describes the count (non-normalized empirical distribution) of the state transitions that occurred from state i, as determined by $\mathbf{q}$. For example, $\boldsymbol{\psi}_i = (2, 1, 0, 0, \ldots, 0)$ means that two transitions occurred from state i to state 1, and one transition occurred from state i to state 2. More formally, let $\mathcal{A}_{i,j} = \{t \mid q_t = i, q_{t+1} = j\}$, let $|\mathcal{A}_{i,j}|$ denote the size of the set $\mathcal{A}_{i,j}$, and let $\mathbf{e}_l^L$ denote an identity vector of length L whose lth element is 1. Then

$$\boldsymbol{\psi}_i = \sum_{j=1}^{L} |\mathcal{A}_{i,j}| \mathbf{e}_j^L. \tag{9}$$

The element γ is a scalar that is the joint log probability of the observations $\mathbf{o}$ and the state sequence $\mathbf{q}$, i.e.,

$$\gamma = \log P(\mathbf{o}, \mathbf{q} \mid \theta) = \log \left(\zeta_{q_T}(o_T) \prod_{t=1}^{T-1} \zeta_{q_t}(o_t) a_{q_t, q_{t+1}} \right). \tag{10}$$

Fig. 1 illustrates the transformation method. The observation sequence $\mathbf{o} = (2, 4, 3, 1, 2, 1, 3, 4)$ is transformed using a 3 state HMM with a codebook of 4 symbols. The most likely state sequence in this example is assumed to be $\mathbf{q} = (1, 1, 1, 2, 2, 3, 3, 3)$.

As we will show, the suggested transformation allows us to discriminatively adjust the score of the discrete HMM system using the SVM technique. We proceed by rearranging the log HMM parameters in a vector form, denoted by $\mathbf{w}_\theta$,

$$\mathbf{w}_\theta = (\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_L, \boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_L) \tag{11}$$

where the kth element of $\boldsymbol{\Phi}_i$ is

$$\Phi_{i,k} = \log \zeta_i(k) \quad k = 1, \ldots, S \tag{12}$$

and the jth element of $\boldsymbol{\Psi}_i$ is

$$\Psi_{i,j} = \log a_{i,j} \quad j = 1, \ldots, L. \tag{13}$$

Using the above notation, we can express the HMM score for $\mathbf{o}$ and $\mathbf{q}$, $\log P(\mathbf{o}, \mathbf{q} \mid \theta)$, in terms of $\mathbf{w}_\theta$ and $\mathbf{T}$. Let $\hat{\mathbf{T}}$ denote the vector $\mathbf{T}$ without its last element. Recall that the last element of $\mathbf{T}$ was denoted by γ, so that

$$\mathbf{T} = (\hat{\mathbf{T}}, \gamma). \tag{14}$$

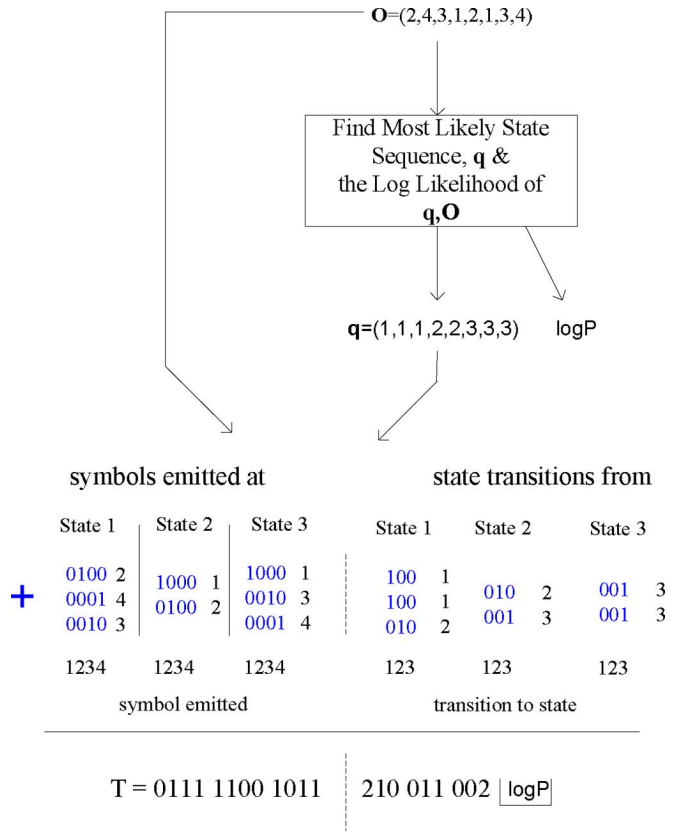

Fig. 1. An example of the variable to fixed length data transformation. In this example we consider a 3 state HMM with a codebook of four symbols.

We can, therefore, write

$$\begin{aligned} \gamma &= \log P(\mathbf{o}, \mathbf{q} \mid \theta) \\ &= \sum_{t=1}^{T-1} \left(\log \zeta_{q_t}(o_t) + \log a_{q_t, q_{t+1}} \right) + \log \zeta_{q_T}(o_T) \\ &= \langle \mathbf{w}_\theta, \hat{\mathbf{T}} \rangle. \end{aligned} \tag{15}$$

Now, in our problem we have M different classes, represented by M corresponding HMMs with parameter vectors θ_m, $m = 1, \ldots, M$. The Viterbi algorithm (in a Bayesian setting) estimates the unknown class $\hat{m}$ using (6), which can also be written as

$$\begin{aligned} \hat{m} &= \arg\max_m \log \left(p_m P(\mathbf{o}, \mathbf{q}_m \mid \theta_m) \right) \\ &= \arg\max_m \langle \mathbf{w}_{\theta_m}, \hat{\mathbf{T}}_m \rangle + b_{\theta_m} \end{aligned} \tag{16}$$

where $\mathbf{q}_m$ is the most likely state sequence corresponding to $\mathbf{o}$, according to the mth HMM, p_m is the prior probability of class m, $b_{\theta_m} = \log p_m$ and

$$\hat{\mathbf{T}}_m = \hat{\mathbf{T}}_m(\mathbf{o}, \mathbf{q}_m). \tag{17}$$

The standard recognizer, (16), can be viewed as the following two stage recognition process. In the first stage, for each model $m = 1, \ldots, M$, we obtain the most likely state sequence $\mathbf{q}_m$, and use it to form $\hat{\mathbf{T}}_m$. In the second stage, for each model m, we make a decision based on the set of scores $\langle \mathbf{w}_{\theta_m}, \hat{\mathbf{T}}_m \rangle + b_{\theta_m}$, $m = 1, \ldots, M$. These scores are obtained by m linear

classifiers with parameters $(\mathbf{w}_{\theta_m}, b_{\theta_m})$ that are functions of the HMM parameters.

In order to improve on the standard recognizer, our first proposal is to modify only the second stage of the recognition process, by using a different set of linear classifiers, with parameters $(\mathbf{w}_m, b_m)$, $m = 1, 2, \ldots, M$, that are obtained by an SVM training approach. Since unlike ML training, the SVM training is discriminative, the new approach is likely to improve the recognition rate. Our classifier applies the following recognition rule:

$$\hat{m} = \arg\max_m S_{\mathbf{w}_m, b_m}(\mathbf{T}_m)$$

where

$$\begin{aligned} S_{\mathbf{w},b}(\mathbf{T}) &\triangleq \langle \mathbf{w}, \mathbf{T} \rangle + b = \langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle + w\gamma + b \\ &= \langle \hat{\mathbf{w}} + w\mathbf{w}_\theta, \hat{\mathbf{T}} \rangle + b. \end{aligned}$$

In the second transition we used (14) and the similar decomposition $\mathbf{w} = (\hat{\mathbf{w}}, w)$, where w is a scalar. In the last transition, we used (15). Thus we can express the SVM score as

$$S_{\mathbf{w},b}(\mathbf{T}) = \langle \tilde{\mathbf{w}}_\theta, \hat{\mathbf{T}} \rangle + \tilde{b}_\theta$$

where

$$\tilde{\mathbf{w}}_\theta = \hat{\mathbf{w}} + w\mathbf{w}_\theta \quad , \quad \tilde{b}_\theta = b. \tag{18}$$

The SVM score can thus be regarded as an adjustment of the baseline HMM score. We can regard the elements of $\hat{\mathbf{w}}$ as tuning values for the HMM log parameters in $\mathbf{w}_\theta$, and w as a scaling parameter. The adjusted parameters described in (18) correspond to an unnormalized HMM, with the following set of parameters. Let us decompose $\hat{\mathbf{w}}$ into two types of elements, similar to (11) as follows:

$$\hat{\mathbf{w}} = \left(\overline{\mathbf{\Phi}}_1, \ldots, \overline{\mathbf{\Phi}}_L, \overline{\mathbf{\Psi}}_1, \ldots, \overline{\mathbf{\Psi}}_L \right). \tag{19}$$

Then by (11), (12), (13), (18), and (19), the vector $\tilde{\mathbf{w}}_\theta$ corresponds to the following unnormalized transition and output probabilities

$$\begin{aligned} \tilde{a}_{i,j} &= \exp(\overline{\Psi}_{i,j} + w \log a_{i,j}) \\ &\qquad i = 1, 2, \ldots, L, \quad j = 1, \ldots, L \qquad (20) \\ \tilde{\zeta}_i(k) &= \exp(\overline{\Phi}_{i,k} + w \log \zeta_i(k)) \\ &\qquad i = 1, \ldots, L, \quad k = 1, \ldots, S. \qquad (21) \end{aligned}$$

Similarly, the scalar b corresponds to the unnormalized prior probability

$$\tilde{p} = e^b. \tag{22}$$

Note that unlike a standard HMM, the unnormalized output and transition probabilities of our unnormalized HMM do not necessarily sum up to one, i.e., $\sum_j \tilde{a}_{i,j}$ and $\sum_k \tilde{\zeta}_i(k)$ are not necessarily one. On the other hand, the prior probabilities of the different models can be renormalized, since this renormalization is equivalent to subtracting a constant from the score of each model. Also note that if we set $\hat{\mathbf{w}} = \mathbf{0}$, $b = \log p$ and $w = 1$, then we return to the standard HMM Viterbi score for $\mathbf{o}$. Thus, the new model generalizes the baseline HMM model. While in standard ML training it is essential to require a valid normalized HMM, when using a discriminative training approach such as SVM training, this normalization condition is not required any more. In fact the unnormalized HMM can be viewed as a generalization of a plain HMM since it represents a wider family of models, and by proper training it can achieve improved recognition results.

Having defined the variable to fixed length data transformation, we proceed to describe two possibilities for training the SVM parameters.

C. Training the SVM Models Using the One Against All Method

The first step in training the SVM models is to transform the training set using the baseline HMM system as was described in Section III-B. Each observation is transformed using all HMMs. Let $\mathcal{O} = (\mathbf{o}^1, \ldots, \mathbf{o}^N)$ and $\mathcal{C} = (c^1, \ldots, c^N)$ be a set of time series observations and their corresponding labels, i.e., $c^n \in \mathcal{W}$ where $\mathcal{W} = \{1, \ldots, M\}$ is the set of classes. $\mathcal{O}$ and $\mathcal{C}$ comprise the training set. Let $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_M)$ denote the set $\mathcal{O}$ transformed using all HMMs. $\mathcal{T}_m$ denotes the transformation of $\mathcal{O}$ using the mth HMM. Let $\mathbf{T}_m^n$ denote $\mathbf{o}^n$ transformed using the mth HMM, so that $\mathcal{T}_m = (\mathbf{T}_m^1, \ldots, \mathbf{T}_m^N)$. Since we are dealing with a multiclass problem, we can use one of the multiclass approaches described in Section II, the one against all method or the transformation to the one class method [13]. We proceed to describe the application of both methods to our problem in detail.

The one against all method, as explained in Section II, trains each of the SVM models separately, but unlike standard ML training, it uses both the positive and the negative examples for training each model. In training SVM model m, the parameters of which we denote by $(\mathbf{w}_m, b_m)$, we use the utterances transformed by HMM model m, denoted by $\mathcal{T}_m$. The SVM label vector $\mathbf{c}_m = (c_m^1, \ldots, c_m^N)$ is a vector whose elements are either 1 or -1 depending on whether the corresponding utterance belongs to model m or not

$$c_m^n = \begin{cases} 1 & \text{if } c^n = m \\ -1 & \text{if } c^n \neq m. \end{cases}$$

The optimization problem for model m is

$$\begin{aligned} \min_{\xi_n, \mathbf{w}_m, b_m} & \frac{1}{2}\|\mathbf{w}_m\|^2 + C \sum_{n=1}^{N} \xi_n \\ \text{s.t.} \quad & c_m^n(\langle \mathbf{w}_m, \mathbf{T}_m^n \rangle + b_m) \geq 1 - \xi_n, \; n = 1, \ldots, N \\ & \xi_n \geq 0, \quad n = 1, \ldots, N. \end{aligned}$$

Each model m is trained so it will tend to assign a positive score to the utterances that belong to model m, and it will tend to assign a negative score otherwise.

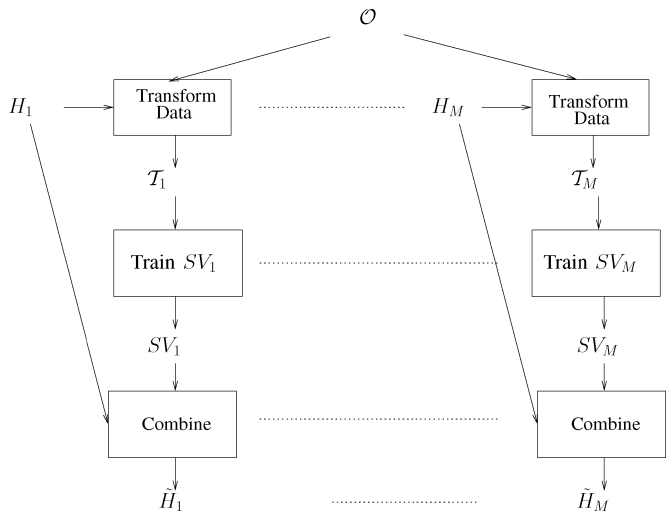

Fig. 2. The one-against-all training method. H_m denotes HMM model m, and $\tilde{H}_m$ denotes unnormalized HMM model m. SV_m denotes SVM model m. First, the data is transformed using all HMM models. Each SVM model is trained using the data transformed by the corresponding HMM model. Finally, each HMM is combined with the corresponding SVM model to form a new unnormalized HMM.

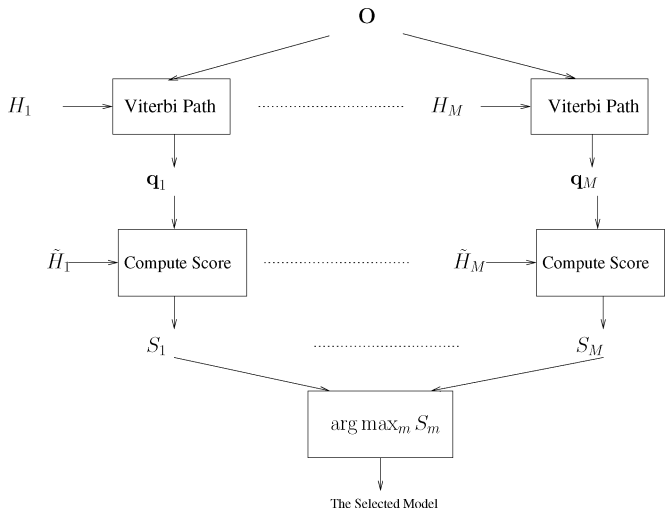

Fig. 3. The 2-HMM recognition process. H_m denotes HMM model m, and $\tilde{H}_m$ denotes unnormalized HMM model m. SV_m denotes SVM model m. One HMM is used to find the most likely state sequence of observation $\mathbf{o}$ and the other is used to compute the score of the state sequence.

We proceed to describe the recognition process. Given an unknown observation $\mathbf{o}$, and M HMM and SVM models trained using the one against all method, a straight forward algorithm for recognition is the following.

1) Find the set $(\mathbf{q}_1, \ldots, \mathbf{q}_M)$ of most likely state sequences corresponding to utterance $\mathbf{o}$ using the baseline HMMs.
2) Compute the vector transformations $\mathbf{T}_m = \mathbf{T}_m(\mathbf{o}, \mathbf{q}_m)$, $m = 1, \ldots, M$.
3) Use the following decision rule to choose the model that best matches the observation

$$\arg\max_m \left\{ \langle \tilde{\mathbf{w}}_m, \mathbf{T}_m \rangle + \tilde{b}_m \right\}.$$

However, in order to make the recognition process as similar as possible to the standard HMM method, we can represent the rescoring SVM models as an unnormalized HMM, as described in Section III-B. The recognition algorithm using the unnormalized HMM set is as follows.

1) Find the set $(\mathbf{q}_1, \ldots, \mathbf{q}_M)$ of most likely state sequences of utterance $\mathbf{o}$ using the baseline HMMs.
2) Compute the log likelihood of utterance $\mathbf{o}$ and the set of most likely state sequences $(\mathbf{q}_1, \ldots, \mathbf{q}_M)$, where $\mathbf{q}_m = q_{m,1}, \ldots, q_{m,T}$, using the unnormalized HMMs. The decision rule is shown in the equation at the bottom of the page, (the superscript m in $\tilde{\zeta}^m_{\cdot}$ and $\tilde{a}^m_{\cdot}$ denotes that the output and transition probabilities of model m should be used).

We refer to this recognition method as the *2-HMM recognition method*. Figs. 2 and 3 summarize the one-against-all training method and the 2-HMM recognition method.

At this point it seems reasonable to try and use the unnormalized HMMs that we obtained to resegment the training database, then to retrain a new set of unnormalized HMMs using the resegmented data, and to proceed iteratively. Unfortunately, empirical evidence (see Section V) shows that when the one against all method is used, the unnormalized HMMs cannot in general be used for finding the best state sequence (i.e., they cannot be used for segmenting the data). On the other hand, the one class transformation training method described below, typically does yield unnormalized HMMs that can be used for segmentation, that is recognition can be done using the unnormalized HMM set in the Viterbi recognizer as if they were plain HMMs.

D. Training the SVM Models Using the One Class Transformation Method

As explained in Section II, the one class transformation method [13] trains all SVM models together, using the entire

$$\hat{m} = \arg\max_m \left\{ \sum_{t=1}^{T-1} \left(\log \tilde{\zeta}^m_{q_{m,t}}(o_t) + \log \tilde{a}^m_{q_{m,t}, q_{m,t+1}} \right) + \log \tilde{\zeta}^m_{q_{m,T}}(o_T) + \log \tilde{p}_m \right\}$$

training set $\mathcal{T}$ along with the correct label set $\mathcal{C}$. The optimization problem is

$$\begin{aligned}\min_{\xi,\mathbf{w},b} & \frac{1}{2}\sum_{m=1}^{M}(\|\mathbf{w}_m\|^2+b_m^2)+C\sum_{n,j,m}\xi_{n,j,m}\\ \text{s.t.}\quad & \langle\mathbf{w}_m,\mathbf{T}_m^n\rangle+b_m-(\langle\mathbf{w}_j,\mathbf{T}_j^n\rangle+b_j)\geq 1-\xi_{n,j,m}\\ & n=1,\ldots,N,\ m=c^n,\ j\in\{1,\ldots,N\}\setminus m\\ & \xi_{n,j,m}\geq 0, n=1,\ldots,N, m=c^n, j\in\{1,\ldots,N\}\setminus m.\end{aligned}$$

All the models are trained simultaneously in an attempt to make the score given by model m to some transformed utterance $\mathbf{T}_m^n$ that belongs to the model (i.e., $c^n=m$), higher than that given to the same utterance transformed by other models.

The use of the trained SVMs for recognition can be done using the 2-HMM recognition method that was described above: The SVM models along with the HMM models are combined into a new set of unnormalized HMMs using (20)–(22). The HMM set is used for segmentation and the unnormalized HMM set is used for scoring. However, as was observed empirically, when using the one class training method, the unnormalized HMMs can typically also be successfully used in the standard Viterbi recognition procedure as if they were plain HMMs. We refer to this recognition procedure as the *1-HMM recognition method*. The 1-HMM recognition method makes it possible to extend the training algorithm into an iterative one, where the new unnormalized HMMs found in one step, are used for segmentation in the next step. The iterative algorithm we propose is the following.

1) Start with the set $\mathcal{T}^0$, which is the set of utterances transformed by the baseline HMM set and train a set of SVMs.
2) Combine the set of SVMs with the set of HMMs used in the previous step into a set of unnormalized HMMs (20)–(22).
3) Use the set of unnormalized HMMs found in the previous step to create a new set of transformed vectors $\mathcal{T}^i$ (8)–(10).
4) Go back to step 1 with the new set $\mathcal{T}^i$.

This approach resembles the segmental K-means algorithm that iteratively segments the data and re-estimates the HMM parameters. The fact that our new unnormalized HMMs can be used for segmentation facilitates the incorporation of our algorithm into existing systems, since no changes are required in the recognition stage. Fig. 4 summarizes the one class transformation method. Recognition can be performed using the 2-HMM approach (one HMM or unnormalized HMM for segmentation and another unnormalized HMM for scoring), as shown in Fig. 3, or by using the 1-HMM approach (only one unnormalized HMM for both segmentation and scoring).

E. Discussion and Relation to Previous Work

The one against all training method has the advantage that it is computationally less demanding than the one class method. On the other hand, the performance of the one class method is usually better, since its criterion accurately expresses our goal, that the score of the correct model would be higher as much as possible than the score of all other models. The goal of the one against all method is to achieve a high positive score for positive examples and a high negative score for negative examples. This goal may be too difficult to achieve, and should be regarded as a sufficient condition for proper classification, but not a necessary one. That is, even if this goal cannot be achieved, good classification may be achieved using the less demanding criterion of the one class method.

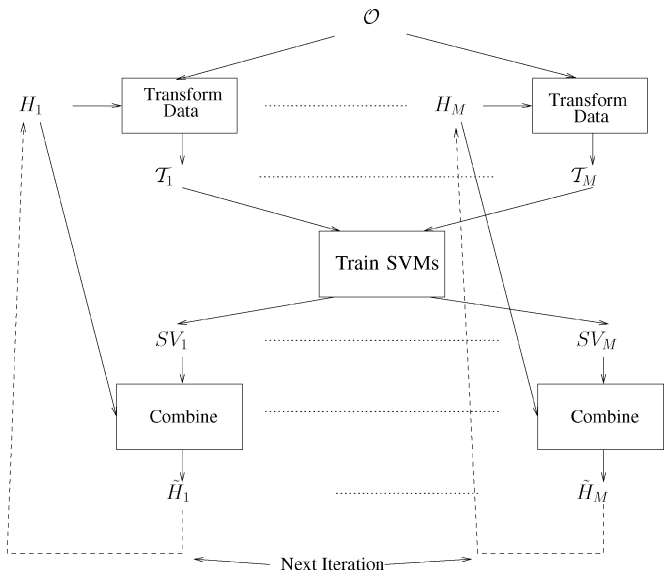

Fig. 4. The one class transformation training method. First, the data is transformed using all HMM models. All SVM models are trained using all the transformed data. Finally, each HMM is combined with the corresponding SVM model to form a new unnormalized HMM.

Our main assertion in this paper is that one class (noniterative) training with 2-HMM recognition improves the performance on the training database, since our classifier is a strict generalization of the standard HMM, and the criterion used in the training is the actual recognition objective. If the training set is sufficiently large, so that there is no over-fitting, then we also expect improvements on the test. Although iterative training may further improve performance, this additional improvement is not guaranteed, neither is the convergence of the iterations. This is due to the fact that the new trained unnormalized HMM may not be suitable for segmenting the data. We note, however, that the iterative training is expected to work better when using the one class method, since by attempting to set the parameters of the classifier, such that positive examples would get a high positive score and negative examples would get a high negative score, the one against all method requires more than is necessary to obtain a good recognizer, and usually needs to shift the HMM parameters far away from their original values that yielded an initial good segmentation. On the hand, the goal of the one class method can usually be achieved by a relatively small shift from the original HMM parameter set. Thus the new parameter set can sometimes still be good enough for segmenting the data.

We now show how our new transformation relates to the Fisher score, used in the Fisher kernel [1]. The Fisher score is defined as

$$\nabla_\theta \log P(\mathbf{o}|\theta).$$

Our transformation can be expressed as follows:

$$\mathbf{T}=(\exp(\mathbf{w}_\theta)\circ\nabla_\theta\log P(\mathbf{o},\mathbf{q}|\theta),\log P(\mathbf{o},\mathbf{q}|\theta))$$

where $\circ$ is the element-wise product between two vectors, θ is the HMM parameter set, and $\log P(\mathbf{o},\mathbf{q}|\theta)$ is defined in (15). Since the classifiers we use and the SVM training scheme involve only linear functions of $\mathbf{T}$, this is equivalent to using

$$\mathbf{T} = (\nabla_\theta \log P(\mathbf{o},\mathbf{q}|\theta), \log P(\mathbf{o},\mathbf{q}|\theta)).$$

Thus we are essentially using a modified Fisher kernel with $\log P(\mathbf{o},\mathbf{q}|\theta)$ replacing $\log P(\mathbf{o}|\theta)$, and with the additional element $\log P(\mathbf{o},\mathbf{q}|\theta)$. By (15) this element is a linear function of $\hat{\mathbf{T}}$ and thus it can be eliminated. However it is included for convenience. Recall that we can achieve at least the same performance as the baseline system. The function $\log P(\mathbf{o},\mathbf{q}|\theta)$ can be represented using the summation (15), unlike the much more complicated function $\log P(\mathbf{o}|\theta)$. Consequently, in spite of the close relationship between our kernel and the Fisher kernel, the development in Section III-B that motivates our method as a discriminative training improvement to the HMM score [see the discussion following (17)], cannot be applied to motivate the Fisher kernel. In addition, the representation of the new model as an unnormalized HMM cannot be applied to the Fisher kernel.

IV. SVM Rescoring of Continuous HMMs

In this section, we present an extension of our algorithm to continuous output probability HMMs. The algorithm uses the following transformation, which is similar to the one presented in Section III-B.

A. A Variable to Fixed Length Data Transformation

Let $\mathbf{o} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T)$ be some observed sequence, such that $\mathbf{o}_t \in \mathbb{R}^d$. Also consider a mixture of Gaussians output probability HMM with L states and G mixtures, and with a parameter set denoted by θ. The parameter set θ is comprised of transition probabilities $a_{i,j}$, $i,j \in \{1,\ldots,L\}$, and the mixture weights, mean vectors and diagonal covariances of the Gaussians, $c_{i,k}$, $\boldsymbol{\mu}_{i,k}$ and $\Lambda_{i,k} = diag(\sigma^2_{i,k,1},\ldots,\sigma^2_{i,k,d})$, $i \in \{1,\ldots,L\}$, $k \in \{1,\ldots,G\}$. We denote

$$\zeta_{i,k}(\mathbf{x}) = c_{i,k} \cdot \frac{1}{\sqrt{(2\pi)^d |\Lambda_{i,k}|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{i,k})\Lambda_{i,k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{i,k})^T\right\} \tag{23}$$

($\mathbf{x}$ and $\boldsymbol{\mu}_{i,k}$ are row vectors). Consider the state and mixture sequence $\tilde{\mathbf{u}} = (\tilde{\mathbf{q}},\tilde{\mathbf{g}})$, where $\tilde{\mathbf{q}} = (\tilde{q}_1,\ldots,\tilde{q}_T)$ and $\tilde{\mathbf{g}} = (\tilde{g}_1,\ldots,\tilde{g}_T)$. Note that $(\mathbf{o},\tilde{\mathbf{u}})$ is the complete data used in the Baum-Welch algorithm. The probability that the HMM assigns to $(\mathbf{o},\tilde{\mathbf{u}})$ is

$$P(\mathbf{o},\tilde{\mathbf{u}} \mid \theta) = \left(\prod_{t=1}^{T-1} \zeta_{\tilde{q}_t,\tilde{g}_t}(\mathbf{o}_t) a_{\tilde{q}_t,\tilde{q}_{t+1}}\right) \cdot \zeta_{\tilde{q}_T,\tilde{g}_T}(\mathbf{o}_T).$$

The probability that this HMM assigns to $\mathbf{o}$ is obtained by summing over all possible state and mixture sequences, $\tilde{\mathbf{u}}$,

$$\begin{aligned} P(\mathbf{o} \mid \theta) &= \sum_{\tilde{\mathbf{u}}} \left(\prod_{t=1}^{T-1} \zeta_{\tilde{q}_t,\tilde{g}_t}(\mathbf{o}_t) a_{\tilde{q}_t,\tilde{q}_{t+1}}\right) \cdot \zeta_{\tilde{q}_T,\tilde{g}_T}(\mathbf{o}_T) \\ &= \sum_{\tilde{\mathbf{q}}} \left(\prod_{t=1}^{T-1} \sum_g \zeta_{\tilde{q}_t,g}(\mathbf{o}_t) a_{\tilde{q}_t,\tilde{q}_{t+1}}\right) \cdot \sum_g \zeta_{\tilde{q}_T,g}(\mathbf{o}_T). \end{aligned}$$

Let $\mathbf{u} = (\mathbf{q},\mathbf{g})$, where $\mathbf{q} = (q_1,\ldots,q_T)$ and $\mathbf{g} = (g_1,\ldots,g_T)$, denote the most likely state and mixture sequence corresponding to $\mathbf{o}$ according to this HMM, i.e.

$$\mathbf{u} = \arg\max_{\tilde{\mathbf{u}}} P(\mathbf{o},\tilde{\mathbf{u}} \mid \theta).$$

We now describe a transformation that yields a new vector $\mathbf{T} = \mathbf{T}(\mathbf{o},\mathbf{u})$ from $\mathbf{o}$ and $\mathbf{u}$. The vector $\mathbf{T}$, whose length is $L^2+LG+dLG+1$, is composed of the vectors $\boldsymbol{\psi}_i, i = 1,\ldots,L$, the vectors $\boldsymbol{\pi}_i$, $i = 1,\ldots,L$, the vectors $\boldsymbol{\phi}_{i,k}$, $i = 1,\ldots,L$, $k = 1,\ldots,G$ and the scalar γ.

$$\mathbf{T} = (\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_L,\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_L,\boldsymbol{\phi}_{1,1},\ldots,\boldsymbol{\phi}_{L,G},\gamma). \tag{24}$$

As in the discrete case, the vector $\boldsymbol{\psi}_i$ describes the count (non-normalized empirical distribution) of the state transitions that occurred from state i, as determined by $\mathbf{q}$, and is defined by (9).

The vector $\boldsymbol{\pi}_i$ describes the count (non-normalized empirical distribution) of the mixtures that were traversed according to $\mathbf{u}$ and belong to state i. For example, $\boldsymbol{\pi}_i = (0,1,3,0,\ldots,0)$ means the most likely state and mixture sequence $\mathbf{u}$ contains four instances of state i, one of which with mixture 2 and the other three with mixture 3. More formally, let $\mathcal{C}_{i,k} = \{t \mid q_t = i, g_t = k\}$, let $|\mathcal{C}_{i,k}|$ denote the size of the set $\mathcal{C}_{i,k}$, and let $\mathbf{e}_k^G$ denote an identity vector of length G whose kth element is 1, then

$$\boldsymbol{\pi}_i = \sum_{k=1}^{G} |\mathcal{C}_{i,k}| \mathbf{e}_k^G. \tag{25}$$

The elements $\boldsymbol{\phi}_{i,k}$ are elements of length d that are used to capture information regarding the means of the kth mixture in state i. Let $\mathcal{C}_{i,k} = \{t \mid q_t = i, g_t = k\}$, then

$$\boldsymbol{\phi}_{i,k} = \sum_{t \in \mathcal{C}_{i,k}} \frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{i,k})\Lambda_{i,k}^{-1} \tag{26}$$

($\mathbf{o}_t$ and $\boldsymbol{\mu}_{i,k}$ are row vectors).

The element γ is a scalar that is the joint log probability of the observations $\mathbf{o}$ and state and mixture sequence $\mathbf{u}$, i.e.,

$$\begin{aligned} \gamma &= \log P(\mathbf{o},\mathbf{u}|\theta) \\ &= \log\left(\left(\prod_{t=1}^{T-1} \zeta_{q_t,g_t}(\mathbf{o}_t) a_{q_t,q_{t+1}}\right) \zeta_{q_T,g_T}(\mathbf{o}_T)\right). \end{aligned} \tag{27}$$

Now assume we have M different classes and M corresponding HMMs. The parameter set of the mth HMM is

denoted by θ_m. The Viterbi algorithm (in a Bayesian setting) estimates the unknown class $\hat{m}$ using the following rule

$$\hat{m} = \arg\max_m \log\left(p_m P(\mathbf{o}, \mathbf{u}_m|\theta_m)\right) \tag{28}$$

where $\mathbf{u}_m$ is the most likely state and mixture sequence corresponding to $\mathbf{o}$, according to the mth HMM, and p_m is the prior probability of class m.

The standard recognizer, (28) can be viewed as the following two stage recognition process. In the first stage, for each model $m = 1, \ldots, M$, we obtain the most likely state and mixtures sequence $\mathbf{u}_m$. In the second stage, for each model m, we make a decision based on the set of scores $\log\left(p_m P(\mathbf{o}, \mathbf{u}_m|\theta_m)\right), m = 1, \ldots, M$.[1]

Using (27) and (23), we can write the HMM score for $\mathbf{o}$ and $\mathbf{u}$ as follows:

$$\begin{aligned}
&\log P(\mathbf{o}, \mathbf{u}|\theta) + \log p \\
&= \sum_{t=1}^{T-1} \log a_{q_t, q_{t+1}} + \sum_{t=1}^{T} \log \zeta_{q_t, g_t}(\mathbf{o}_t) + \log p \\
&= \sum_{t=1}^{T-1} \log a_{q_t, q_{t+1}} + \sum_{t=1}^{T} \log c_{q_t g_t} \\
&\quad + \sum_{t=1}^{T} \left(-\frac{1}{2} \left(\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t}\right) \Lambda^{-1}_{q_t, g_t} (\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t})^T \right) \\
&\quad - \sum_{t=1}^{T} K_{q_t, g_t} + \log p
\end{aligned} \tag{29}$$

where

$$K_{q_t, g_t} \triangleq \log\left((2\pi)^d |\Lambda_{q_t, g_t}|\right)^{1/2}.$$

In order to improve on the standard recognizer we propose to use a set of linear classifiers, with parameters $(\mathbf{w}_m, b_m)$, $m = 1, \ldots, M$, that are obtained by an SVM training approach. Our classifier applies the following recognition rule:

$$\hat{m} = \arg\max_m S_{\mathbf{w}_m, b_m}(\mathbf{T}_m)$$

where

$$\begin{aligned}
S_{\mathbf{w},b}(\mathbf{T}) &\triangleq \langle \mathbf{w}, \mathbf{T} \rangle + b \\
&= \langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle + w\gamma + b \\
&= \langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle + w \log P(\mathbf{o}, \mathbf{u}|\theta) + b.
\end{aligned} \tag{30}$$

Let us decompose $\hat{\mathbf{w}}$ into three types of elements, similar to (19) in the discrete case, as follows:

$$\hat{\mathbf{w}} = (\overline{\boldsymbol{\Psi}}_1, \ldots, \overline{\boldsymbol{\Psi}}_L, \overline{\boldsymbol{\Pi}}_1, \ldots, \overline{\boldsymbol{\Pi}}_L, \overline{\boldsymbol{\Phi}}_{1,1}, \ldots, \overline{\boldsymbol{\Phi}}_{1,G}, \ldots, \overline{\boldsymbol{\Phi}}_{L,G}). \tag{31}$$

The claims below motivate our new method.

[1]A variant on the above rule is to obtain only the most likely state sequence of the mth model, $\mathbf{q}_m$, and then to make a decision based on $\log\left(p_m P(\mathbf{o}, \mathbf{q}_m|\theta_m)\right)$, for $m = 1, \ldots, M$.

Claim 4.1: The SVM score given in (30) can be viewed as the HMM score given in (29) with a modified set of unnormalized HMM parameters

$$\begin{aligned}
S_{\mathbf{w},b}(\mathbf{T}) &= \langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle + w \log P(\mathbf{o}, \mathbf{u}|\theta) + b \\
&= \sum_{t=1}^{T-1} \log \tilde{a}_{q_t, q_{t+1}} + \sum_{t=1}^{T} \log \tilde{c}_{q_t g_t} \\
&\quad + \sum_{t=1}^{T} \left\{ -\frac{1}{2} w \left(\mathbf{o}_t - \tilde{\boldsymbol{\mu}}_{q_t, g_t}\right) \Lambda^{-1}_{q_t, g_t} \left(\mathbf{o}_t - \tilde{\boldsymbol{\mu}}_{q_t, g_t}\right)^T \right\} \\
&\quad - w \sum_{t=1}^{T} K_{q_t, g_t} + \log \tilde{p}
\end{aligned} \tag{32}$$

where $\mathbf{w} = (\hat{\mathbf{w}}, w)$,

$$\tilde{a}_{i,j} = \exp(\overline{\Psi}_{i,j} + w \log a_{i,j}) \quad i = 1, \ldots, L, \quad j = 1, \ldots, L \tag{33}$$

$$\tilde{c}_{i,k} = \exp(\overline{\Pi}_{i,k} + A_{i,k} + w \log c_{i,k}) \quad i = 1, \ldots, L, \quad k = 1, \ldots, G \tag{34}$$

$$\tilde{\boldsymbol{\mu}}_{i,k} = \frac{\left(\frac{1}{w}\overline{\boldsymbol{\Phi}}_{i,k} + 2\boldsymbol{\mu}_{i,k}\right)}{2} \quad i = 1, \ldots, L, \ k = 1, \ldots, G \tag{35}$$

$$\tilde{\sigma}^2_{i,k,r} = \sigma^2_{i,k,r} \quad i = 1, \ldots, L, \ k = 1, \ldots, G, \ r = 1, \ldots, d \tag{36}$$

$$\tilde{p} = e^b \tag{37}$$

$$\begin{aligned}
A_{i,k} &= -\frac{1}{2} w \left(\frac{1}{w}\overline{\boldsymbol{\Phi}}_{i,k} + \boldsymbol{\mu}_{i,k}\right) \Lambda^{-1}_{i,k} \boldsymbol{\mu}^T_{i,k} \\
&\quad + \frac{1}{2} w \left(\frac{\frac{1}{w}\overline{\boldsymbol{\Phi}}_{i,k} + 2\boldsymbol{\mu}_{i,k}}{2}\right) \Lambda^{-1}_{i,k} \left(\frac{\frac{1}{w}\overline{\boldsymbol{\Phi}}_{i,k} + 2\boldsymbol{\mu}_{i,k}}{2}\right)^T.
\end{aligned} \tag{38}$$

We prove the claim in Appendix I.

$S_{\mathbf{w},b}(\mathbf{T})$ can thus be interpreted as the score of an unnormalized HMM with parameters $\tilde{a}_{i,j}$, $\tilde{c}_{i,k}$, $\tilde{\boldsymbol{\mu}}_{i,k}$, $\tilde{\sigma}^2_{i,k,r}$, and $\tilde{p}$. Recall that in a continuous unnormalized HMM the transition probabilities do not necessarily sum up to one, and the Gaussian mixture models do not necessarily integrate to one. In fact, in the continuous case we have an additional degree of freedom, since each Gaussian density function is raised to the power of w. If we set $w = 1$ for all models then we obtain a standard unnormalized HMM.

The SVM models can be trained as described in Sections III-C and III-D. The following claim asserts that our method can produce any variance-constrained unnormalized HMM (i.e., an arbitrary unnormalized HMM, except for its variance components, that are identical to those of the given HMM). The implication is that our method yields the variance-constrained unnormalized HMM that yields the best discrimination, either in the sense of the one against all method or in the sense of the one class transformation method.

Claim 4.2: Consider an arbitrary HMM defined by $a_{i,j}$, $c_{i,k}$, $\boldsymbol{\mu}_{i,k}$, $\sigma^2_{i,k,r}$, and p, where $i, j \in \{1, \ldots, L\}$, $k \in \{1, \ldots, G\}$, and $r \in \{1, \ldots, d\}$. Also consider an unnormalized HMM defined by $\tilde{a}_{i,j}$, $\tilde{c}_{i,k}$, $\tilde{\boldsymbol{\mu}}_{i,k}$, $\sigma^2_{i,k,r}$ and $\tilde{p}$ (i.e., it is an arbitrary unnormalized HMM, except that it has the same variance parameters as the given HMM). Then there exists a vector $\mathbf{w} = (\hat{\mathbf{w}}, w = 1)$

such that (33)–(37) transform the given HMM to the given unnormalized HMM.

Proof: The claim is proved by setting

$$\begin{aligned}\overline{\mathbf{\Phi}}_{i,k} &= 2(\tilde{\boldsymbol{\mu}}_{i,k} - \boldsymbol{\mu}_{i,k}) \\ \overline{\Pi}_{i,k} &= \log \tilde{c}_{i,k} - \log c_{i,k} - A_{i,k} \\ \overline{\Psi}_{i,j} &= \log \tilde{a}_{i,j} - \log a_{i,j} \\ \text{and} \quad b &= \log \tilde{p}\end{aligned}$$

where $A_{i,k}$ is given in (38). ∎

Note that in the discrete HMM case a similar claim applies: The transformation defined by (20)–(22) can yield an arbitrary unnormalized HMM. Hence, in the discrete case the training produces the best unnormalized HMM in the sense of the one against all or the one class transformation method.

B. Relation to Previous Work

As in the discrete case, we proceed to show how the suggested transformation (24) relates to the Fisher score. Recall that the Fisher score is defined as

$$\nabla_\theta \log P(\mathbf{o}|\theta).$$

Our transformation can be expressed as follows:

$$\mathbf{T} = \left(\left(a_{1,1}, \ldots, a_{L,L}, c_{1,1}, \ldots, c_{L,G}, \frac{1}{2}, \ldots, \frac{1}{2} \right) \circ \nabla_{\bar{\theta}} \log P(\mathbf{o}, \mathbf{u}|\theta), \log P(\mathbf{o}, \mathbf{u}|\theta) \right)$$

where $\circ$ is the element-wise product between two vectors, $\bar{\theta}$ is the HMM parameter set excluding the covariance matrices, and

$$\log P(\mathbf{o}, \mathbf{u}|\theta) = \sum_{t=1}^{T-1} \left(\log \zeta_{q_t, g_t}(\mathbf{o}_t) + \log a_{q_t, q_{t+1}} \right) + \log \zeta_{q_T, g_T}(\mathbf{o}_T).$$

Since the classifiers we use and the SVM training scheme involve only linear functions of $\mathbf{T}$, this is equivalent to using

$$\mathbf{T} = \left(\nabla_{\bar{\theta}} \log P(\mathbf{o}, \mathbf{u}|\theta), \log P(\mathbf{o}, \mathbf{u}|\theta) \right).$$

V. Experiments

In this section, we describe experiments conducted using our algorithm on a toy problem and on an isolated noisy digit recognition task, and compare the results to the standard ML trained HMM system. Both discrete and continuous HMM models are considered. Note that although continuous HMMs typically yield better results than discrete HMMs in the task of speech recognition, discrete HMMs are computationally more efficient.

A. Toy Problem

Our continuous HMM algorithm was first applied to a toy problem where there is a model mismatch, to demonstrate the benefit of our approach under model mismatch conditions. We used three continuous HMMs with 5 states and 2 mixtures per state, as underlying distributions for three classes.

TABLE I
The Results of the One-Class Training Method. $C = 2^{-7}$

Set	Baseline	2-HMM Recognition Rate	Improvement in Error Rate
Training set	97.67%	99.78%	90.56%
Test set	96.89%	99.22%	74.91%

The transition probability matrix of each HMM was left to right, such that when the process is in state i, it can either remain in that state or skip to the next state, $i+1$. The self transition probabilities were determined by drawing them at random with uniform probability in the range [0,1], and then if the drawn value, p, was less than 0.5, it was reset to $1-p$, such that all transition probabilities were set in the range [0.5,1]. The last state is an absorbing state, that is its self transition probability was 1. The resulting self transition probabilities of states 1–4 of the first HMM were 0.7689, 0.7604, 0.8729, and 0.5134. The state transition probabilities of states 1–4 of the second HMM were 0.9257, 0.8407, 0.5159, and 0.8689, and the state transition probabilities of states 1–4 of the third HMM were 0.6119, 0.9456, 0.6919, and 0.9056. The feature vector was 26-dimensional. The output vector in each state was distributed as a mixture of two Gaussians, with mean vector components that were chosen at random, statistically independent of the other components, such that $50 \leq \mu_{i,1} \leq 150$ and $-150 \leq \mu_{i,2} \leq -50$, where $\mu_{i,k}$ is the mean of some kth mixture component at state i. Similarly, each variance component of the Gaussians was chosen at random, statistically independent of the other components, using a uniform distribution in [0,10]. We note that the qualitative behavior of our results did not change much when the experiment was repeated with another realization of HMM parameters.

Each HMM was used to generate a training set of 300 samples and a test set of 300 samples. The three classes were then modeled using three 5 state HMMs with a single mixture at each state. The HMM parameters were estimated using the training set and the ML approach. The parameters were then adjusted using our continuous one-class transformation training algorithm, and the 2-HMM recognition method. The parameter C was chosen through a process of 10-fold cross validation from the set $(2^{-10}, 2^{-9}, 2^{-8}, 2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1})$. The results are presented in Table I.

As a comparison, when there was no model mismatch, and the three classes were modeled using three 5 state HMMs with two mixtures, the recognition rate was 100% both on the training and on the test data. Thus this example demonstrates that under mismatch conditions, where our model is far from the true one, our new approach can significantly improve the recognition rate (74.91% improvement on the test set).

B. The TIDIGITS Database

The TIDIGITS corpus [26] is a multispeaker isolated and continuous digit vocabulary database of 326 speakers. It con-

sists of 11 words, "1" through "9" plus "oh" and "zero." In our experiments, we used only the isolated speech part of the database. The training set we used in our experiments was comprised of 112 speakers, 55 men and 57 women. Each digit was uttered twice by each speaker, so we had a total of 224 utterances for each digit. Our test set was comprised of 113 speakers, 56 men and 57 women, and a total of 226 utterances per digit.

Isolated digit recognition on this database using the standard Gaussian mixture HMM yields very high recognition rates (close to a 100%). We therefore added white Gaussian noise with variance equal to the signal power, obtaining a low, 0 dB signal-to-noise ratio (SNR).

C. Discrete HMMs

The baseline discrete HMM speech recognition system was trained using the HTK toolkit [27]. At the first stage, feature extraction was performed on the training and test sets. The feature vector was comprised of 12 Mel-frequency cepstral coefficients, a log energy coefficient and the corresponding delta coefficients, for a total of 26 coefficients. The frame rate was 10 ms with a 25 ms window size. The feature vectors extracted from the training set were used to create a linear codebook of 150 symbols with a diagonal covariance Mahalanobis distance metric. The training and test data were then transformed into discrete symbol sequences, and 11 left-to-right discrete HMM models were trained using the quantized training set. Each discrete HMM model contained 10 emitting states and two non-emitting entry and exit states. The HMMs were trained using 8 segmental K-means iterations for parameter initialization, followed by 15 Baum-Welch iterations. The recognition rate using this system was 89.18% on the test set and 94.85% on the training set.

The discrete HMM parameters obtained using the maximum likelihood estimation were used as the baseline system. We tested our algorithm with both the one-against-all SVM training method and the one class transformation SVM training method. We used the hidden Markov toolbox for Matlab [28] and the probabilistic model toolkit (PMT) [29] to work with the unnormalized HMM and SVM models.

1) The One Against All Method: The SVM models were trained using the OSU SVM toolbox [30]. The value of the parameter C for each of the models was chosen from the set $(2^{-9}, 2^{-8}, 2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1})$ through a fivefold cross-validation process. The training data was partitioned into five sets. Each time a different set was used as the test set and a model was trained using the other four sets. The cross validation recognition rate was defined as the average recognition rate on all five sets. C was set to the value that yielded the highest cross validation recognition rate. The selected values of C for classes $1, \ldots, 11$ were $(2^{-7}, 2^{-7}, 2^{-6}, 2^{-6}, 2^{-7}, 2^{-9}, 2^{-9}, 2^{-8}, 2^{-7}, 2^{-8}, 2^{-7})$, respectively.

After the value of C was selected for a particular model, the training process was repeated using that value and all the training data. The SVM models and the baseline HMMs were combined to form unnormalized HMM models (20)–(22). When the unnormalized HMMs were used as if they were plain HMMs in the Viterbi recognizer (1-HMM recognition), the recognition rate did not show an improvement compared to the baseline system. The 2-HMM recognition method gave a 27.81% recognition rate improvement compared to the baseline system. The results are summarized in Table II.

TABLE II
THE RESULTS OF APPLYING THE ONE-AGAINST-ALL TRAINING METHOD TO DISCRETE HMMS

Set	Baseline	2-HMM Recognition Rate	Improvement
Training set	94.85%	98.25%	66%
Test set	89.18%	92.19%	27.81%

2) The One Class Transformation Method: The SVM models were trained using the libSVM toolbox [31] with some modifications. C was chosen from the set $(2^{-13}, 2^{-12}, 2^{-11}, 2^{-10}, 2^{-9}, 2^{-8}, 2^{-7}, 2^{-6})$ through a tenfold cross-validation process. The training data was partitioned into ten sets. Each time a different set was used as the test set and all models were trained using the other nine sets. C was set to the value that yielded the best cross validation recognition rate when using the 1-HMM recognition method (i.e., when the unnormalized HMM was used in the Viterbi recognizer). We then trained the models using all the training data and the same value of C. The SVM and HMM models were combined to form unnormalized HMMs. We tested the recognition rate both using the 1-HMM recognition method and the 2-HMM recognition method. We continued the process iteratively, using the unnormalized HMM set to resegment the training data at each iteration, for a total of 30 iterations. The results of the first and last iterations are presented in Table III. The rest are shown in Fig. 5. The graphs show the recognition rate on the test and train data using the 1 and 2 HMM recognition methods. Both graphs slightly fluctuate, and are in general increasing. The recognition rates using both methods are close and coincide from iteration 17 on. The recognition rate on the test set increases and eventually fluctuates around 93.3%.

D. Continuous HMMs

We conducted experiments using a single mixture baseline system and a 5 mixture baseline system. The baseline speech recognition systems were trained using the HTK toolkit [27]. At the first stage, feature extraction was performed on the training and test sets. The feature vector was comprised of 12 Mel-frequency cepstral coefficients, a log energy coefficient and the corresponding delta and acceleration coefficients, for a total of 39 coefficients. Cepstral mean normalization was applied. The frame rate was 10 ms with a 25 ms window size. The training set was used to produce 11 left-to-right single mixture continuous HMM models, and 11 left-to-right, 5 mixture continuous HMM models. Each HMM model contained 10 emitting states and two non-emitting entry and exit states. A diagonal covariance matrix was used. The single mixture HMMs were trained by using 3 segmental K-means iterations for parameter initialization, followed by 7 Baum-Welch iterations. The 5 mixture HMM models were trained by first producing 11 single mixture HMMs, initialized using 3 segmental K-means iterations. The number of mixtures at each state was then incremented by 1, by splitting the mixture with the largest mixture weight, and then

TABLE III
THE FIRST AND LAST ITERATIONS USING THE ONE CLASS TRAINING METHOD FOR A DISCRETE HMM. THE PARAMETER C WAS SELECTED THROUGH A TENFOLD CROSS-VALIDATION PROCESS

Set	Baseline	2-HMM Recognition	Improvement	1-HMM Recognition	Improvement
Training set, It. 1	94.85%	98.37%	68.34%	98.25%	62.14%
Training set, It. 30	–	99.75%	95.14%	99.75%	95.14%
Test set, It. 1	89.18%	92.39%	29.667%	92.43%	30%
Test set, It. 30	–	93.32%	38.26%	93.32%	38.26%

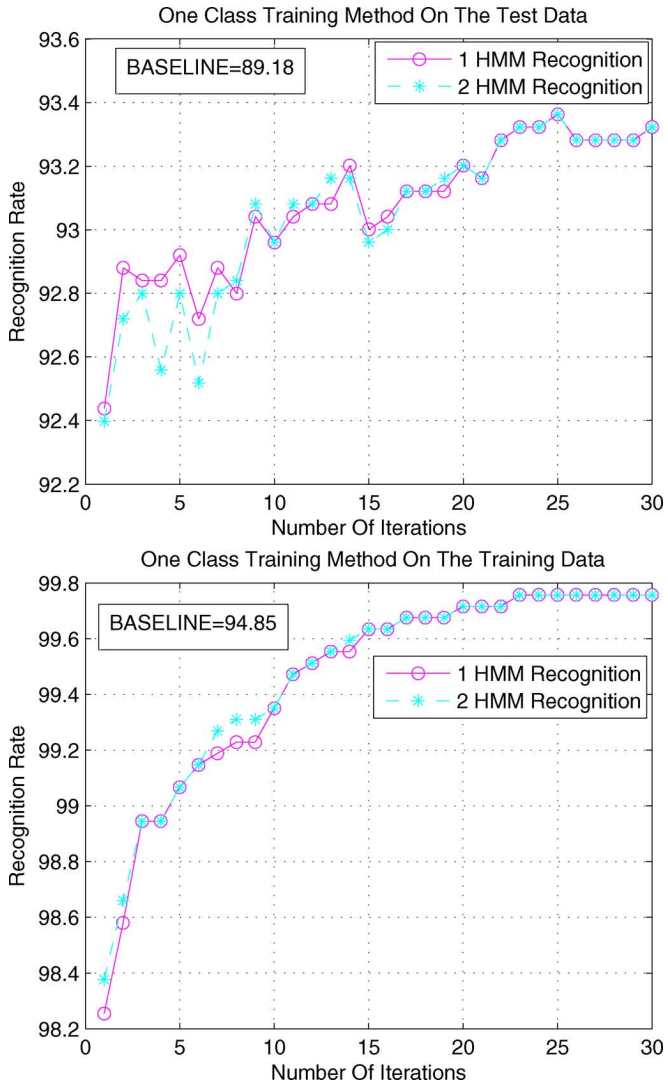

Fig. 5. The results of the one class training method in the discrete HMM case.

by reestimating the parameters using 7 Baum-Welch iterations. The process was repeated until 5 mixture models were obtained. The recognition rate using the single mixture Gaussian system was 88.58% on the test set and 91.59% on the training set. The recognition rate on the 5 Gaussian system was 92.75% on the test set and 97.52% on the training set.

We used the single mixture system to test both the one-against-all SVM training method and the one class transformation SVM training method. The 5 mixture system was only used to test the one class transformation SVM training method using the 2-HMM recognition method. We used the hidden Markov toolbox for Matlab [28] and the probabilistic model toolkit (PMT) [29] to work with the unnormalized HMM models. In all the experiments described below, the training data was normalized so that all vector elements were in the interval $[-1,1]$. This normalization was applied due to numerical reasons.

1) The One Against All Method: The SVM models were trained using the OSU SVM toolbox [30]. The parameter C of each word model was chosen from the set $2^{(-12,-11,-10,-9,-8,-7,-6,-5,-4,-3,-2,-1)}$ through a fivefold cross-validation process. The results using the 2-HMM recognition method were 73.25% recognition rate on the test set and 76.17% on the training set. In light of that, no further experiments were conducted using the one-against-all training method. As was explained in Section III-E, the one class transformation method is typically better than one against all.

2) The One Class Transformation Method, Single Mixture Models: The SVM models were trained using the libSVM toolbox [31] with some modifications. First, C was selected from the set

$$(2^{-12}, 2^{-11}, 2^{-10}, 2^{-9}, 2^{-8}, 2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}).$$

The training data was partitioned into two sets consisting of 90% and 10% of the data. The SVM models were trained using 90% of the data with each possible value of C, and the system was tested on the 10% cross validation data using the 2-HMM recognition method. The value that maximized the cross validation recognition rate was selected. After choosing C, the SVM models were trained using the entire training set, and tested using the 2-HMM recognition method. The baseline system was evaluated using both the Viterbi algorithm and the forward algorithm, and both algorithms yielded very similar results.

Recall that in the continuous HMM case, each Gaussian density function is raised to the power of w. In Table IV, we present the results of our method with 2-HMM recognition, both when the parameter w of each model can attain an arbitrary value, and when the w parameters of all models are forced to be equal. In the latter case, the w parameters of the SVMs are tied together, and thus after the training they can all be normalized to one (multiplying the vectors $\mathbf{w}$ of the SVMs by a positive constant does not affect the recognition results). Since parameter

TABLE IV
THE RESULTS OF THE ONE CLASS TRAINING METHOD IN THE SINGLE MIXTURE CONTINUOUS HMM CASE. THE PARAMETER C WAS SELECTED THROUGH A CROSS VALIDATION PROCESS. THE VALUE OF C FOR THE TIED SYSTEM WAS 2^{-1} AND THE VALUE OF C FOR THE UNTIED SYSTEM WAS 2^{-3}

Set	Baseline (Viterbi Algorithm)	Baseline (Forward Algorithm)	2 HMM Recognition, untied SVMs	Improvement	2 HMM tied SVMs	Improvement
Training	91.59%	91.63%	98.58%	83.11%	99.31%	91.79%
Test	88.53%	88.54%	93.44%	42.55%	93.48%	42.90%

tying did not affect the results, we continued our experiments with the tied SVM system.

When we tried to use the 1-HMM approach, we observed a significant performance loss. The following iterative method produced an unnormalized HMM that can be successfully used in the 1-HMM recognition operation mode. Although this recognizer is not as good as the 2-HMM recognizer that we start with, the advantage of 1-HMM recognition is that it uses the standard recognition algorithm (the Viterbi algorithm). Thus, we are able to significantly improve on the baseline by only replacing the parameters of our HMM (from the normalized baseline HMM to the unnormalized reestimated HMM). From (33)–(37), we see that if we replace $\mathbf{w} = (\hat{\mathbf{w}}, w = 1)$ by $(\beta\hat{\mathbf{w}}, w = 1)$ where β is some constant that satisfies $0 < \beta < 1$, then the new β-*normalized* unnormalized HMM will be shifted closer to the original HMM. Thus for β sufficiently small, we would be able to use the β-normalized unnormalized HMM also for segmenting the data, i.e., it would be possible to apply the 1-HMM recognition method successfully.

We continued our experiments by fixing C to the value that was used to produce the results of Table IV (tied SVMs). We continued the training iteratively, using the unnormalized HMM created at each step for segmentation in the next step. The recognition results at each iteration were measured both for the β-normalized SVM and for the non-normalized SVM ($\beta = 1$). We used 6 iterations, and β was selected from the set $(0.001, 0.001 + \Delta, \ldots, 0.1)$, $\Delta = 0.0025$ as follows. The training set was partitioned into seven sets that constitute 70%,5%,5%,5%,5%,5%,5% of the data. At the first iteration, 70% of the data was used to train the SVM models using the selected value of C and to create unnormalized HMM models using all possible values of β. The value of β was chosen so as to maximize the 1-HMM recognition rate using the unnormalized HMM on 5% of the training data. After β was selected for the first iteration, 75% of the data was used to train the SVM models and derive the unnormalized HMMs to be used in the second iteration. The process was repeated until 6 values of β, one for each iteration, were chosen. The training process was then done using the entire training set and the recognition rate at each iteration was evaluated. The selected values of β were (0.0135, 0.0385, 0.026, 0.001, 0.026, 0.006). The results are presented in Fig. 6 (the baseline results are indicated by a horizontal line). In each graph and each iteration, the SVM training is conducted by using the segmented data produced by the unnormalized HMM that we have at the beginning of the iteration. In the 2-HMM recognition, no β case, β normalization is not applied after training the SVMs. As can be seen, the results of the 2-HMM recognition method without β normalization are generally higher than the results achieved using β normalization. However, the use of β-normalization enables the application of the standard recognition method (1-HMM), with a significant error rate reduction compared to the baseline system.

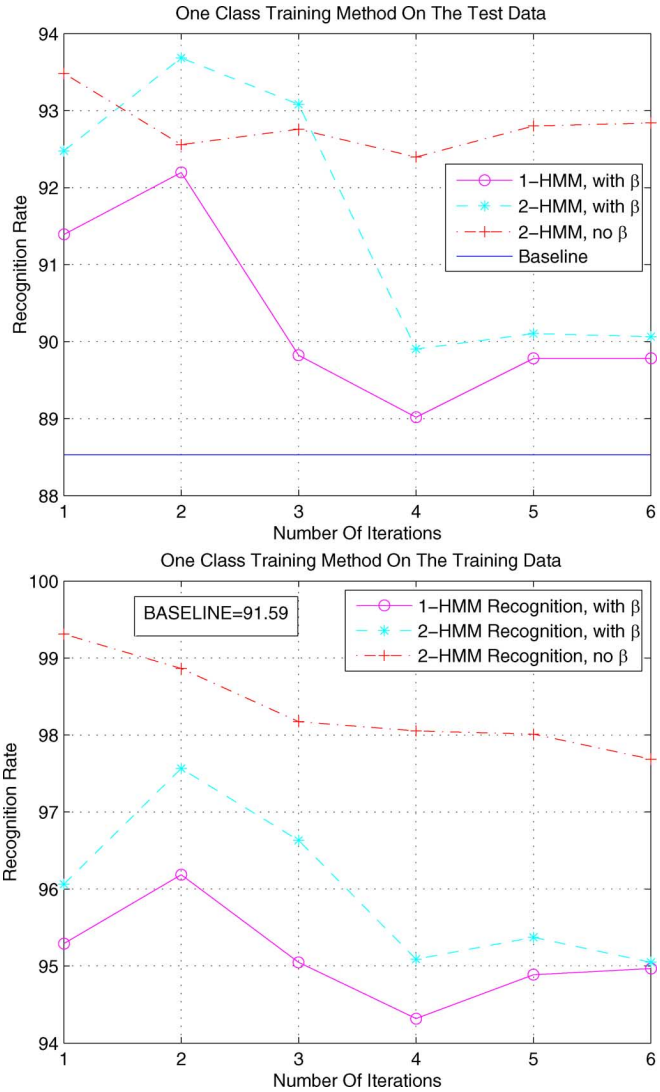

Fig. 6. The results of the one class training method for single mixture continuous HMMs.

3) The One Class Transformation Method, Five Mixture Models: We trained the unnormalized HMM models using the 5 mixture HMM models as a baseline system (with both the Viterbi and forward algorithms) and the one class transformation method. We tested the performance of the system

TABLE V
THE RESULTS OF THE ONE CLASS TRAINING METHOD FOR FIVE MIXTURES CONTINUOUS HMMS AT $\mathrm{SNR} = 0$ dB

Set	Baseline (Viterbi Algorithm)	Baseline (Forward Algorithm)	2 HMM Recognition, untied SVM parameters	Improvement
Training	97.52%	97.56%	99.87%	91.79%
Test	92.75%	92.96%	94.93%	30.06%

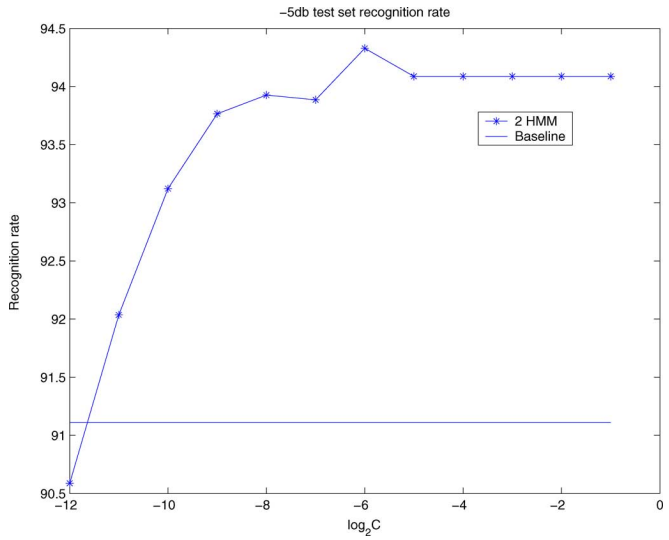

Fig. 7. The results of the one class training method on the test database for five mixtures continuous HMMs at $\mathrm{SNR} = -5$ dB.

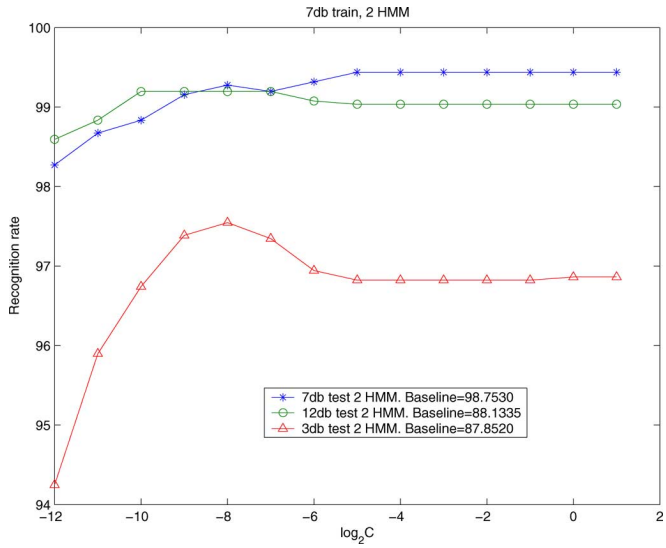

Fig. 8. The results of the one class training method on the test database for five mixtures continuous HMMs. Training was performed at $\mathrm{SNR} = 7$ dB.

using 2-HMM recognition (i.e, the baseline HMMs were used for segmentation and the unnormalized HMMs were used for scoring). We used cross validation to determine the value of the parameter C, and set it to $C = 2^{-7}$. The results are summarized in Table V.

The above experiment was repeated on the same database, using the same algorithms, except that the SNR was changed to -5 dB. Fig. 7 presents the results on the test database for different values of the parameter C. The baseline results are also shown. As can be seen, the results are robust to the value of C over a wide range of values. As in the previous experiments, a cross-validation database can be used to estimate a good value of C. The maximum improvement of our method compared to the baseline is 36.2% reduction in the error rate. The baseline performance on the training database is 95.8% correct, while our method yields 100% correct in the range $2^{-6} \leq C \leq 1$.

Another aspect of our new approach is that it provides more robustness to the estimated model. This property is in agreement with the fact that SVM training searches for the hyperplane with the best separation between positive and negative training examples. To demonstrate this attribute of our approach, we trained the five mixture HMM system on the same isolated part of the TIDIGITS database at $\mathrm{SNR} = 7$ dB, using the standard (baseline) and new (one class, 2-HMM mode) training methods. We then tested the performances of the resulting recognition systems at SNRs of 3, 7, and 12 dB. Fig. 8 presents the results for different values of the parameter C. As can be seen, the new method yields a much better robustness to SNR mismatch between the train and test conditions. At $\mathrm{SNR} = 3$ dB the baseline performance is 87.85% recognition rate on the test, while our new method yields 97.54% recognition rate for the optimal C. At $\mathrm{SNR} = 12$ dB the baseline performance is 88.13% recognition rate on the test, while our new method yields 99.2% recognition rate for the optimal C. As a comparison, we have also evaluated the performance of the baseline under optimal training conditions (i.e., without mismatch) and obtained the following. When we train at $\mathrm{SNR} = 3$ dB the recognition results on the test data at the same SNR conditions ($\mathrm{SNR} = 3$ dB) are 97.55% correct. When we train at $\mathrm{SNR} = 12$ dB the recognition results on the test data at the same SNR conditions ($\mathrm{SNR} = 12$ dB) are 99.48% correct. Thus our new method significantly improves the robustness of the trained system in an SNR region around that used in the training, and brings it close to (and sometimes even beyond) the matched training results. We note that when the mismatch is larger, the improvement of our method compared to the baseline degrades.

VI. CONCLUSION

In this paper, we presented the SVM rescoring of hidden Markov models algorithm. The algorithm offers a discriminative training scheme that utilizes the SVM technique to rescore the results of an ML trained baseline discrete or continuous HMM system. The rescoring model can be represented as an unnormalized HMM. The unnormalized HMM can be viewed as a generalization of a plain HMM since it represents a wider family of models, and by proper training it can achieve improved recognition results.

We started by describing the variable to fixed length data transformation that uses the most likely path, as determined by the baseline HMM system, to transform variable length data into fixed length data vectors. We then presented two methods for training the SVM models, one of which was extended to an iterative algorithm similar to the segmental K-means algorithm. We explained how the baseline HMMs can be combined with the trained SVM models to create a set of unnormalized HMMs. Two recognition methods were presented: 1-HMM recognition, that uses only the unnormalized HMM set as if it were a set of plain HMMs, and 2-HMM recognition, that uses the baseline HMM set for segmentation and the unnormalized HMM set to rescore the results of the baseline HMM set. We described the algorithm for both discrete and continuous output probability HMMs.

We assessed the performance of our algorithm on a toy problem and on an isolated noisy digit recognition task. We tested both training methods and both recognition algorithms and compared them to the standard ML trained system for both the discrete and continuous cases. We observed significant reduction in word error rate. One class noniterative training and 2-HMM recognition yielded a significant improvement in the recognition rate, both for discrete and for continuous HMMs. The iterative one class training algorithm yielded further improvements in the discrete HMM case.

There are several issues that were not dealt with in this paper and require further research. We have restricted our attention to isolated speech recognition. An extension to continuous speech recognition can be achieved using 1-HMM recognition by combining the unnormalized HMMs into composite unnormalized HMMs. Another possible extension can be achieved using 2-HMM recognition by using an N-best list (a list of N most likely paths) with SVM rescoring.

Another straight forward extension of our algorithm is training the unnormalized HMM models with parameter tying. This can be done using the one class SVM training method.

Another possibility is to modify the continuous HMM transformation by computing the derivatives with respect to the variance elements as well. The transformation will then be

$$\mathbf{T} = (\nabla_\theta \log P(\mathbf{o}, \mathbf{u}|\theta), \log P(\mathbf{o}, \mathbf{u}|\theta))$$

instead of

$$\mathbf{T} = (\nabla_{\bar{\theta}} \log P(\mathbf{o}, \mathbf{u}|\theta), \log P(\mathbf{o}, \mathbf{u}|\theta))$$

where θ is the HMM parameter set and $\bar{\theta}$ is the HMM parameter set excluding the covariance elements.

APPENDIX
PROOF OF CLAIM 4.1

Proof: Using (24) and (31) we have

$$\langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle = \sum_{i=1}^{L} \langle \overline{\boldsymbol{\Psi}}_i, \boldsymbol{\psi}_i \rangle + \sum_{i=1}^{L} \langle \overline{\boldsymbol{\Pi}}_i, \boldsymbol{\pi}_i \rangle + \sum_{i=1}^{L} \sum_{k=1}^{G} \langle \overline{\boldsymbol{\Phi}}_{i,k}, \boldsymbol{\phi}_{i,k} \rangle$$

and using (9), (25), and (26) we get

$$\langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle = \sum_{i=1}^{L} \left\langle \overline{\boldsymbol{\Psi}}_i, \sum_{j=1}^{L} |\mathcal{A}_{i,j}| \mathbf{e}_j^L \right\rangle + \sum_{i=1}^{L} \left\langle \overline{\boldsymbol{\Pi}}_i, \sum_{k=1}^{G} |\mathcal{C}_{i,k}| \mathbf{e}_k^G \right\rangle + \sum_{i=1}^{L} \sum_{k=1}^{G} \left\langle \overline{\boldsymbol{\Phi}}_{i,k}, \sum_{t \in \mathcal{C}_{i,k}} \frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_{i,k}) \Lambda_{i,k}^{-1} \right\rangle. \quad (39)$$

Recall that $\mathbf{e}_i^L$ is a vector of length L whose ith element is 1 and the rest are 0. Therefore, (39) is equivalent to

$$\langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle = \sum_{i=1}^{L} \sum_{j=1}^{L} |\mathcal{A}_{i,j}| \overline{\Psi}_{i,j} + \sum_{i=1}^{L} \sum_{k=1}^{G} |\mathcal{C}_{i,k}| \overline{\Pi}_{i,k} + \sum_{i=1}^{L} \sum_{k=1}^{G} \left\langle \overline{\boldsymbol{\Phi}}_{i,k}, \sum_{t \in \mathcal{C}_{i,k}} \frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_{i,k}) \Lambda_{i,k}^{-1} \right\rangle.$$

Recall that $\mathcal{A}_{i,j} = \{t \mid q_t = i, q_{t+1} = j\}$ and $\mathcal{C}_{i,k} = \{t \mid q_t = i, g_t = k\}$. Thus

$$\begin{aligned}
\langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle &= \sum_{i=1}^{L} \sum_{j=1}^{L} |\{t \mid q_t = i, q_{t+1} = j\}| \overline{\Psi}_{i,j} \\
&\quad + \sum_{i=1}^{L} \sum_{k=1}^{G} |\{t \mid q_t = i, g_t = k\}| \overline{\Pi}_{i,k} \\
&\quad + \sum_{i=1}^{L} \sum_{k=1}^{G} \left\langle \overline{\boldsymbol{\Phi}}_{i,k}, \sum_{t \in \{t \mid q_t = i, g_t = k\}} \frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_{i,k}) \Lambda_{i,k}^{-1} \right\rangle \\
&= \sum_{t=1}^{T-1} \overline{\Psi}_{q_t, q_{t+1}} + \sum_{t=1}^{T} \overline{\Pi}_{q_t, g_t} \\
&\quad + \sum_{t=1}^{T} \left\langle \overline{\boldsymbol{\Phi}}_{q_t, g_t}, \frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t}) \Lambda_{q_t, g_t}^{-1} \right\rangle.
\end{aligned}$$

Plugging this into (30) and then using (29) yields,

$$\begin{aligned}
S_{\mathbf{w},b}(\mathbf{T}) &= \langle \hat{\mathbf{w}}, \hat{\mathbf{T}} \rangle + w \log P(\mathbf{o}, \mathbf{u}|\theta) + b \\
&= \sum_{t=1}^{T-1} \overline{\Psi}_{q_t, q_{t+1}} + \sum_{t=1}^{T} \overline{\Pi}_{q_t, g_t} \\
&\quad + \sum_{t=1}^{T} \left\langle \overline{\boldsymbol{\Phi}}_{q_t, g_t}, \frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t}) \Lambda_{q_t, g_t}^{-1} \right\rangle \\
&\quad + w \log P(\mathbf{o}, \mathbf{u}|\theta) + b \\
&= \sum_{t=1}^{T-1} \overline{\Psi}_{q_t, q_{t+1}} + \sum_{t=1}^{T} \overline{\Pi}_{q_t, g_t} \\
&\quad + \sum_{t=1}^{T} \left\langle \overline{\boldsymbol{\Phi}}_{q_t, g_t}, \frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t}) \Lambda_{q_t, g_t}^{-1} \right\rangle \\
&\quad + w \sum_{t=1}^{T-1} \log a_{q_t, q_{t+1}} + w \sum_{t=1}^{T} \log c_{q_t, g_t} \\
&\quad + \sum_{t=1}^{T} \left\{ -\frac{1}{2} w (\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t}) \Lambda_{q_t, g_t}^{-1} (\mathbf{o}_t - \boldsymbol{\mu}_{q_t, g_t})^T \right\} \\
&\quad - w \sum_{t=1}^{T} K_{q_t, g_t} + b.
\end{aligned}$$

Rearranging terms we get

$$\begin{aligned} S_{\mathbf{w},b}(\mathbf{T}) = & \sum_{t=1}^{T-1}(\overline{\Psi}_{q_t,q_{t+1}} + w\log a_{q_t,q_{t+1}}) \\ & + \sum_{t=1}^{T}(\overline{\Pi}_{q_t,g_t} + w\log c_{q_t,g_t}) \\ & + \sum_{t=1}^{T}\left\{\left\langle \overline{\mathbf{\Phi}}_{q_t,g_t}, \left(\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})\Lambda_{q_t,g_t}^{-1}\right)\right\rangle \right. \\ & \left. - w\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})\Lambda_{q_t,g_t}^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})^T\right\} \\ & - w\sum_{t=1}^{T}K_{q_t,g_t} + b. \end{aligned} \tag{40}$$

Let us now focus on the term inside the third summation on the right hand side and show that it can be interpreted as the tuning of the mixture means.

$$\begin{aligned} & \left\langle \overline{\mathbf{\Phi}}_{q_t,g_t}, \frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})\Lambda_{q_t,g_t}^{-1}\right\rangle \\ & - w\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})\Lambda_{q_t,g_t}^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})^T \\ & = \frac{1}{2}\left(\overline{\mathbf{\Phi}}_{q_t,g_t} - w(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})\right)\Lambda_{q_t,g_t}^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})^T \\ & = -\frac{1}{2}w\left(\mathbf{o}_t - \left(\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + \boldsymbol{\mu}_{q_t,g_t}\right)\right)\Lambda_{q_t,g_t}^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_{q_t,g_t})^T \\ & = -\frac{1}{2}w\left(\mathbf{o}_t - \frac{\left(\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + 2\boldsymbol{\mu}_{q_t,g_t}\right)}{2}\right)\Lambda_{q_t,g_t}^{-1} \\ & \cdot\left(\mathbf{o}_t - \frac{(\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + 2\boldsymbol{\mu}_{q_t,g_t})}{2}\right)^T + A_{q_t,g_t} \end{aligned} \tag{41}$$

where

$$\begin{aligned} A_{q_t,g_t} = & -\frac{1}{2}w\left(\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + \boldsymbol{\mu}_{q_t,g_t}\right)\Lambda_{q_t,g_t}^{-1}\boldsymbol{\mu}_{q_t,g_t}^T \\ & + \frac{1}{2}w\left(\frac{\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + 2\boldsymbol{\mu}_{q_t,g_t}}{2}\right)\Lambda_{q_t,g_t}^{-1}\left(\frac{\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + 2\boldsymbol{\mu}_{q_t,g_t}}{2}\right)^T. \end{aligned}$$

Substituting (41) in (40) and rearranging terms yields

$$\begin{aligned} S_{\mathbf{w},b}(\mathbf{T}) = & \sum_{t=1}^{T-1}\left\{\overline{\Psi}_{q_t,q_{t+1}} + w\log a_{q_t,q_{t+1}}\right\} \\ & + \sum_{t=1}^{T}\left\{\overline{\Pi}_{q_t,g_t} + w\log c_{q_tg_t} + A_{q_t,g_t}\right\} \\ & + \sum_{t=1}^{T}\left\{-\frac{1}{2}w\left(\mathbf{o}_t - \frac{\left(\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + 2\boldsymbol{\mu}_{q_t,g_t}\right)}{2}\right)\Lambda_{q_t,g_t}^{-1}\right. \\ & \left. \cdot\left(\mathbf{o}_t - \frac{\left(\frac{1}{w}\overline{\mathbf{\Phi}}_{q_t,g_t} + 2\boldsymbol{\mu}_{q_t,g_t}\right)}{2}\right)^T\right\} \\ & - w\sum_{t=1}^{T}K_{q_t,g_t} + b. \end{aligned}$$

The definitions (33)–(37), thus, yield (32). ■

REFERENCES

[1] T. Jaakkola, M. Diekhans, and D. Haussler, "A discriminative framework for detecting remote protein homologies," *J. Computat. Biol.*, vol. 7, pp. 95–114, 2000.

[2] N. Smith and M. Gales, *Speech Recognition Using SVMs*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, vol. 14, Adv. Neural Inf. Process. Syst..

[3] V. Wan and S. Renals, "Speaker verification using sequence discriminant support vector machines," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 2, pp. 203–210, Mar. 2005.

[4] C. Bahlmann, B. Haasdonk, and H. Burkhardt, "On-line handwriting recognition with support vector machines – A kernel approach," in *Proc. 8th IWFHR*, 2002, pp. 49–54.

[5] J. Keshet, S. Shalev-Shwartz, Y. Singer, and D. Chazan, "Phoneme alignment based on discriminative learning," in *9th Europ. Conf. Speech Commun. Technol. (INTERSPEECH)*, 2005.

[6] A. Ganapathiraju, J. E. Hamaker, and J. Picone, "Applications of support vector machines to speech recognition," *IEEE Trans. Signal Process.*, vol. 52, pp. 2348–2355, Aug. 2004.

[7] A. Sloin and D. Burshtein, "Support vector machine rescoring of hidden Markov models," presented at the 24th IEEE Conf. Elect. Electron. Eng., Eilat, Israel, Nov. 2006.

[8] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[9] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[10] A. Ng, Cs229 Stanford Lecture Notes 2003 [Online]. Available: http://www.stanford.edu/class/cs229/notes/cs229-notes3.pdf

[11] J. C. Platt, *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*, B. Scholkopf, C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT, 1999, vol. 12, Adv. Kernel Methods – Support Vector Learning, pp. 185–208.

[12] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 415–425, Mar. 2002.

[13] V. Franc and V. Hlavác, "Multi-class support vector machine," in *Proc. 16th Int. Conf. Pattern Recog. (ICPR'02)*, 2002, vol. 2, pp. 236–239.

[14] O. L. Mangasarian and D. R. Musicant, "Successive overrelaxation for support vector machines," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1032–1037, Sep. 1999.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc.*, vol. 39, no. 1, pp. 1–38, 1977.

[16] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*. Englewood Cliffs, NJ: Prentice-Hall PTR, 2001.

[17] L. R. Rabiner, J. G. Wilpon, and B.-H. Juang, "A segmental k-means training procedure for connected word recognition," *AT&T Tech. J.*, pp. 21–40, May-Jun. 1986.

[18] B.-H. Juang and L. R. Rabiner, "The segmental k-means algorithm for estimating parameters of hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 9, pp. 1639–1641, Sep. 1990.

[19] L. R. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall PTR, 1993.

[20] N. Merhav and Y. Ephraim, "Maximum likelihood hidden Markov modeling using a dominant sequence of states," *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2111–2115, Sep. 1991.

[21] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Machine Learn. (ICML)*, MA, Jun. 2001.

[22] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," in *Adv. Neural Inf. Process. Syst. 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[23] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden Markov support vector machines," presented at the 20th Int. Conf. Mach. Learn. (ICML), Washington, DC, Aug. 2003.

[24] L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans, "Discriminative unsupervised learning of structured predictors," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 1057–1064.

[25] W. Xu, J. Wu, and Z. Huang, "A maximum margin discriminative learning algorithm for temporal signals," in *Proc. 18th Int. Conf. Pattern Recogn. (ICPR'06)*, Hong Kong, Aug. 2006, vol. 2, pp. 460–463.

[26] R. G. Leonard, "A database for speaker-independent digit recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 1984, vol. 9, pp. 328–331.

[27] *HTK3 – Hidden Markov Model Toolkit Version 3.2.1*, , 2002 [Online]. Available: http://htk.eng.cam.ac.uk
[28] K. Murphy, Hidden Markov Toolbox for Matlab 1998 [Online]. Available: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
[29] Probabilistic Model Toolkit (PMT) HP Labs [Online]. Available: http://www.hpl.hp.com/downloads/crl/pmt/
[30] J. Ma, Y. Zhao, S. Ahalt, and D. Eads, OSU SVM: A Support Vector Machine Toolbox for Matlab 2001 [Online]. Available: http://svm.sourceforge.net/license.shtml
[31] C.-C. Chang and C.-J. Lin, LIBSVM: A Library for Support Vector Machines 2001 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

Alba Sloin received the B.Sc. degree in electrical engineering and computer science and the M.Sc. degree in electrical engineering in 2003 and 2006, respectively, from Tel-Aviv University, Tel-Aviv, Israel.

Her research interests include information theory, machine learning, and signal processing.

David Burshtein (M'92–SM'99) received the B.Sc. and Ph.D. degrees in electrical engineering in 1982 and 1987, respectively, from Tel-Aviv University, Tel-Aviv, Israel.

During 1988–1989, he was a Research Staff member in the Speech Recognition Group of IBM, T. J. Watson Research Center. In 1989, he joined the School of Electrical Engineering, Tel-Aviv University, where he is presently an Associate Professor. His research interests include information theory and signal processing.