EXPLOITING SUPPORT VECTOR MACHINES IN HIDDEN MARKOV MODELS FOR SPEAKER VERIFICATION

Xin Dong, Wu Zhaohui and Yang Yingchun

Department of Computer Science and Engineering, Zhejiang University, China
{xindong, wzh, yyc}@cs.zju.edu.cn

ABSTRACT

Hidden Markov Models have been proved to be an efficient way for statistically modeling sequence signals. And the Support Vector Machines seem to be a promising candidate to perform the classification task. A new method combining support vector machine and hidden Markov models is proposed. The output of support vector machines are modified as posterior probability using sigmoid function, and act as a probability evaluator in the hidden states of HMM.

1. INTRODUCTION

Speaker Verification is a problem of determining by machine whether or not a voice sample provides sufficient match to a claimed identity [1,2]. Speaker verification systems are either text-dependent or text-independent. We restrict ourselves to the problem of text-independent speaker verification, where the text of speech is arbitrary and unknown in both training and testing.

Many structures have been proposed for speaker verification. The two most popular techniques are Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM) [3,4]. An alternative technique presented in [5,6] use discriminative classifier Support Vector Machines (SVM) for speaker verification. The main advantage of this approach is the classifier is trained discriminatively with an algorithm achieving global optimization.

In this paper, we propose a new text-independent speaker verification method based the combination of Support Vector Machine and Hidden Markov Models. The support vector method is a discriminative approach, modeling the boundaries directly between speakers’ voices in some feature space while keeping discriminative power low [7,8,9,10]. The SVM works very well for single input vectors, however, it is difficult to deal with the sequences of vectors (such as speech). A possible way to incorporate temporal correlation in the SVM is by a Hidden Markov Model (HMM). In such model, the non-stationary speech signal is represented as a sequence of states. In [11], Jaakkola derived kernel function for use in discriminative methods such as support vector machines from generative probability models such as HMM. Here, we embed the support vector machines as probability generator [12,13] into Hidden Markov Models’ states.

In next Section, we review the HMM methods. Section 3 describes our approach to SVM-HMM hybrid. The application on speaker verification is shown in Section 4. Finally, the conclusions of the study are presented.

2. HIDDEN MARKOV MODELS

A Hidden Markov Models is a stochastic function of a Markov chain. As such it is composed of two elements: a Markov process and a set of stochastic function, or output probabilities.

2.1 The Markov Process

The Markov process composed of N symbolic states denoted by \( S = \{S_1, S_2, \ldots, S_N\} \), which are supposed to represent the acoustic clusters of the speech signal, and transition probabilities between these states. We denote by \( q_t \) the actual state at time \( t \), with \( q_t \in S \).

The transitions between states are determined by the matrix \( A = (a_{ij}) \), where
\[
a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i), \quad 1 \leq i, j \leq N
\]

When the model is ergodic we generally mean that all transitions between states are possible. In a more restricted sense, we allow all the transitions in a single step, or \( a_{ij} > 0 \) for all \( i \) and \( j \). An ergodic model is characterized by a unique stationary probability distribution of states, which can be used as the initial probabilities for the modeling, i.e.,
\[
\pi_j = P(q_1 = S_j) = P(q_t = S_j)
\]
for a typical state sequence.
2.2 The Output Probability

To each of the states we attach a probability distribution for the observations, called the output probabilities and are the more important component of the HMM. The observation probability density at state \( j \), is denoted by \( b_j (O_t) \), where

\[
b_j (O_t) = P[O_t \mid q_t = S_j]
\]  

(3)

Where \( O_t \) is an observation of the sample sequence \( O = O_1O_2 \cdots O_T \). In the next section, we will show how to estimate the output probability using SVM method.

3. PROBABILITY OUTPUT FOR SVM

Standard SVMs do not provide probabilities output. However, the outputs of SVM classifiers are not uncalibrated values, using the geometrical interpretation of the classifying hyperplane and the distance of the patterns from the hyperplane, we calculate the posterior probability.

3.1 Support Vector Machine

SVM are classifiers based on the principle of structural risk minimization. Experimental results indicate that SVM can achieve a generalization performance that is greater than or equal to other classifiers, while requiring significantly less training data to achieve such an outcome. The main idea of binary classification Support Vector approach is to construct a hyperplane to separate the two classes (labeled \( y \in \{-1,1\} \)), and let the decision function be:

\[
f(x) = \text{sign}(w \cdot x + b)
\]  

(4)

To maximize the margin (the distance between the hyperplane and the nearest point) gives the following optimization problem: minimize

\[
\phi(w, \xi) = \frac{1}{2} (w \cdot w) + C \sum_{i=1}^{l} \xi_i
\]  

(5)

With constraints

\[
y_i ((w \cdot x_i) + b) \geq 1 - \xi_i, i = 1, 2, \cdots l
\]

\[
\xi_i \geq 0, i = 1, 2, \cdots l
\]

The dual solution to this problem is: maximize the quadratic from (6) under constraints (7).

\[
W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j)
\]  

(6)

With constraints

\[
0 \leq \alpha_i \leq C, i = 1, 2, \cdots l
\]

\[
\sum_{i=1}^{l} \alpha_i y_i = 0
\]  

(7)

Where \( K(x_i, x_j) \) is kernel function.

Giving the decision function:

\[
f(x) = \text{sign} (\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b)
\]  

(8)

3.2 Convert SVM output to posterior probability

The classification of the SVM is given by (4), while the activation is:

\[
g(x) = w \cdot x + b
\]  

(9)

Where \( x \) is an input vector.

The relation is invariant under a positive rescaling of the argument inside (4). Thus a canonical hyperplane is defined so that \( |g(x)| = 1 \) for the closest points (support vectors).

It is clear that for a given hyperplane \( g(x) = 0 \), and for a vector \( x \) that does not belong to the hyperplane, we have:

\[
g(x) = \pm d ||w||
\]  

(10)

where \( d \) is the distance of the point \( x \) to the given hyperplane. The different signs appoint the side of the hyperplane for the vector \( x \). So we can see that the output \( g(x) \) of the SVM is actually the multiplication of the norm of the vector \( w \) and the distance from the chosen hyperplane, and analogously

\[
d_x = \frac{g(x)}{||w||}
\]  

(11)

And the margin between the canonical hyperplane and the closest points (support vectors) is:

\[
d_{sv} = \frac{1}{||w||}
\]  

(12)

Clearly, the ratio of \( d_x \) to \( d_{sv} \) is \( g(x) \). Using the rate of the distance, we propose a modification of the SVM outputs. We convert the output of the SVM to the posterior probability. This estimate is applied to a sigmoid function to yield:

\[
P(C_{+1} \mid x) = \frac{1}{1 + e^{-g(x)}}
\]  

(13)

The sigmoid transforms correct classifications to a quantity roughly equal to one and misclassification to a quantity roughly equal to zero.

3.3 Posterior probability in multi-class case

The following strategies were applied to build M classes classifiers utilizing binary SVM classifiers: one-against-rest classifiers and one-against-one classifiers. The one-
against-one strategy is discussed here. As for the class $i$, M-1 binary classifiers are combined to estimate the final output. Each classifier is trained on a subset of the training examples of the two involved classes (class $i$ and an other class). For this, the training samples have to be relabeled: Members of the $i$th class are labeled to 1; members of the other classes are labeled to –1. In the classification phase, the M-1 different classifiers output M-1 estimated posterior probabilities of the current input vector. And these probabilities constitute a M-1 dimension vector:

$$V(x) = [P_{ia1}(C_i \mid x), \cdots, P_{iaj}(C_i \mid x), \cdots, P_{iaM}(C_i \mid x)]^T$$

(14)

where $P_{iaj}(C_i \mid x)$ is the probability output of the binary SVM for $i$th class and $j$th class, $i \neq j$.

A Gaussian Model is used to estimate the distribution of the probability vector $V(x)$ and output a new posterior probability as:

$$P(C_i \mid x) = N(V(x), \mu, \Sigma) = ((2\pi)^{-d/2} |\Sigma|^{-1}) \exp\left[ -\frac{1}{2} (V(x) - \mu)^T \Sigma^{-1} (V(x) - \mu) \right]$$

(15)

4. TEXT-INDEPENDENT SPEAKER VERIFICATION

Text-independent speaker verification is implemented by modeling each speaker with an individual class. The operation consists of two phases: training and verification. The quality of any model is determined by how well it fits the observed data during the training and the verification phases. When the data is stochastic, or in the absence of complete understanding of the data when we want to treat it as stochastic, only a probabilistic quality measure can be given. The most natural measure in the case is the conditional probability of the observation given the model. That is $P(O \mid \lambda)$, where $O = O_1O_2\cdots O_T$ are the sequences of vectors of samples, $\lambda = (A, B, \pi)$ are the model parameters. $B$ denotes the Gaussian parameters in the probability output of SVM. The better the model describes the data the higher this probability, or likelihood, is.

4.1 The Training Phase

The training problem is thus to find the most probable model parameters, given the data. The first training step is to select the dimensions of the model, i.e., the number of states. Once these are determined, there is an effective dynamic programming algorithm, the forward-backward procedure for HMM to evaluate the estimation probability $P(O \mid \lambda)$. In order to maximize the likelihood, the estimation-maximization (EM) iterative method, via the Baum-Welch reestimation algorithm is used. Due to the large number of local maxima of the likelihood function in the model’s parameter space, in order to converge to a good model iteratively, a carefully selected initial model is needed.

The k-means clustering method is used to construct the initial model. When the states number of HMM is determined, i.e. $N$, the training data of a certain class, i.e. class $i$, is clustered into $N$ states. The initial transitions probabilities hidden markov models for class $i$ are estimated by counting the transitions between states in the training data.

The support vector machines are trained with global discrimination. Now there are total $M$ different classes ($M$ different HMMs), we construct $N$ discriminate multi-class classifiers ($N$ different states) for each class, i.e., class $i$. The members of each state plus the training data of all other classes are used to train the SVM classifier for that state. To reduce the training time, each class data is again clustered into $L$ mixtures. And the resulting $M \times L$ centroids are used for training.

4.2 The Verification Phase

It is easy to see that $\log P(O_T \mid \lambda)$ grows, on the average, linearly with $T$, the log likelihood per frame:

$$\frac{1}{T} \log P(O_T \mid \lambda)$$

(16)

is thus the natural normalized measure between the model $\lambda$ and the data sequence $O_T$. Using the Viterbi algorithm again, we can calculate the log likelihood of the most probable state sequence in an efficient way, linear in $T$. The verification phase is done by comparing the normalized score to a threshold $\xi$.

$$\frac{1}{T} \log P(O_T \mid \lambda) > \xi \quad \text{accept speaker}$$

$$\text{otherwise reject speaker}$$

(17)

4.3 Experiment

We applied our method on the YOHO database [14]. YOHO uses combination lock phrases, e.g., “26-81-57”. The YOHO has four enrollment sessions with 24 phrases per session. Verification consists of 10 sessions with 4 phrases per session. The features were derived using 12th order LPC analysis and deltas (making up a twenty four dimensional feature space) on a 30 milliseconds frame every 10 milliseconds. The frames of data corresponding to silence were removed from the utterance. The first 50 speakers (labeled 101 to 154) are selected in our test. In
support vector training, the number of cluster centroids \( L \) was set as 100 and the RBF kernel was used with \( \sigma = 0.5 \).

<table>
<thead>
<tr>
<th>Number of States</th>
<th>EER (%)</th>
<th>Standard deviation of EER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.07</td>
<td>0.084</td>
</tr>
<tr>
<td>2</td>
<td>0.91</td>
<td>0.067</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>0.052</td>
</tr>
<tr>
<td>4</td>
<td>0.79</td>
<td>0.044</td>
</tr>
<tr>
<td>5</td>
<td>0.94</td>
<td>0.037</td>
</tr>
</tbody>
</table>

Table 1. Verification equal error rate (EER) on YOHO

The experiment results are shown in Table 1, where the verification equal error rates (false accept = false reject) are presented. When the HMM has unique state, the system is a SVM approach. One can see that with the increase of state number, the verification EER achieves better performance. We also find that the best result is achieved when the number of state is 3, after that the improvement of the performance is not obvious. That is because we are focus on the text-independent speaker verification, the temporal information is helpful to the task but not so important.

We also compared our method with the traditional Continuous Density Hidden Markov Models (using Gaussian Mixture Models in the hidden states). The FA (false accept) FR (false reject) curves is shown in Fig. 1. The comparison was made with the same experiment set. And the number of HMM states is 3. One can see that the SVM-HMM hybrid architecture outperformed the CDHMM method.

![Fig 1. FA-FR curves of CDGMM and SVM-HMM hybrid](image)

5. CONCLUSIONS

A new method combining discriminative classifier support vector machine and generative model hidden Markov models is proposed. The outputs of support vector machines are modified as posterior probability using sigmoid and Gaussian combination, and act as a probability evaluator in the hidden states of HMM. We also show the experiment on speaker recognition.

REFERENCES


Acknowledgment: This work was supported by National High Technology Research & Development Programme(863) of China (No. 2001AA4180) and Zhejiang Provincial Natural Science Foundation for Young Scientist (No. RC01058)