AN INITIAL ATTEMPT FOR PHONEME RECOGNITION USING STRUCTURED SUPPORT VECTOR MACHINE (SVM)

Hao Tang[1], *Chao-Hong Meng*[2], *Lin-Shan Lee*[1,2]

[1]Graduate Institute of Electrical Engineering, National Taiwan University
[2]Graduate Institute of Computer Science and Information Engineering, National Taiwan University
larryniven4@gmail.com, mno2.csie@gmail.com, lslee@gate.sinica.edu.tw

ABSTRACT

Structured Support Vector Machine (SVM) is a recently developed extension of the very successful SVM approach, which can efficiently classify structured pattern with maximized margin. This paper presents an initial attempt for phoneme recognition using structured SVM. We simply learn the basic framework of HMMs in configuring the structured SVM. In the preliminary experiments with TIMIT corpus, the proposed approach was able to offer an absolute performance improvement of 1.33% over HMMs even with a highly simplified initial approach, probably because of the concept of maximized margin of SVM. We see the potential of this approach because of the high generality, high flexibility, and high power of structured SVM.

Index Terms— Hidden Markov Model, Structured Support Vector Machine, Phoneme Recognition

1. INTRODUCTION

Hidden Markov models (HMMs) have been almost the only most successful approach for phoneme recognition for a very long time [1]. As the area of machine learning is getting mature, more and more research efforts tries to find new paradigms for the task with a hope that some of them may work better than HMMs in one way or the other. For example, HMMs are known to be generative, while quite several machine learning approaches have better discriminative power.

Large margin HMM is a very successful example along this direction [2]. In this approach, the original expectation maximization (EM) algorithm for parameter estimation in HMM was replaced by the large margin convex optimization. This is similar to the concept of margin maximization in the Support Vector Machine (SVM). Besides, instead of replacing the EM algorithm with large margin optimization, there were also approaches using SVM to capture the Markov assumptions [3]. Conditional Random Field (CRF) [4] is another emerging new approach to the problem, probably because of the similarity between CRF and HMM. It is based on a looser independence assumption, and has produced very successful results as well [5].

On the other hand, an extended form of "structured SVM" has been developed from the very successful SVM approach [6], which can classify the structured patterns efficiently with maximized margin. This structured SVM seems to be able to handle at least the phoneme recognition problem in speech recognition. Therefore, in this work, we present an initial attempt to try to use the structured SVM to handle the phoneme recognition task by simply learning the basic framework from HMMs in configuring the structured SVM. Although this is only an initial attempt with a highly simplified framework, we see its potential due to the high generality, high flexibility, and powerfulness of structured SVM. Also even only with a simplified framework learned from HMMs, the initial experiments with the TIMIT corpus already offered an 1.33% absolute performance improvement as compare to HMMs.

2. PROPOSED APPROACH — PHONEME RECOGNITION USING STRUCTURED SUPPORT VECTOR MACHINE

Here we briefly summarize the structured SVM [6] and explain how it can be used for phoneme recognition.

2.1. Brief Summary of Structured Support Vector Machine

In the context of supervised learning, we are given a set of training instances, $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n) \in X \times Y$, where $\mathbf{x}_i$ is a feature vector, and $\mathbf{y}_i$ is the corresponding correct label, and we wish to assign correct labels to unknown feature vectors. The desired labels $\mathbf{y}_i$ can be not just classes or real numbers, but structured objects, such as sequences, strings, trees, lattices, or graphs.

In order to deal with structured objects as mentioned above, one approach is to define a function $f = f(\mathbf{x}; \mathbf{w}) : X \to Y$, where $\mathbf{w}$ is the parameter vector for the machine to learn. A relatively simple way to design such a desired function f is to give every possible structured label $\mathbf{y}$ for a given feature $\mathbf{x}$ a score by another scoring function $F = F(\mathbf{x}, \mathbf{y}; \mathbf{w}) : X \times Y \to \mathbb{R}$, and simply take the structured label $\mathbf{y}$ giving the highest score as the output of f,

$$f(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in Y} F(\mathbf{x}, \mathbf{y}; \mathbf{w}). \tag{1}$$

Borrowing the maximized margin concept from SVM, we also wish to maximize the margin between the scores of the correct label and the nearest incorrect label. This concept is sketched in a simplified diagram as shown in Fig 1. We can further simplify the scoring function F by requiring it to be linear,

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle, \tag{2}$$

where $\Psi(\mathbf{x}, \mathbf{y})$ is a function representing the structured relationship between $\mathbf{x}$ and $\mathbf{y}$, and $\langle \cdot, \cdot \rangle$ represents inner product. We can then train the parameter vector $\mathbf{w}$ using training instances $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, N$, and then classify the desired label $\mathbf{y}$ for an unknown testing feature vector $\mathbf{x}$ using the scoring function $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ with the trained parameter vector $\mathbf{w}$. Now the problem reduces to how to train the parameter vector $\mathbf{w}$ fast enough.

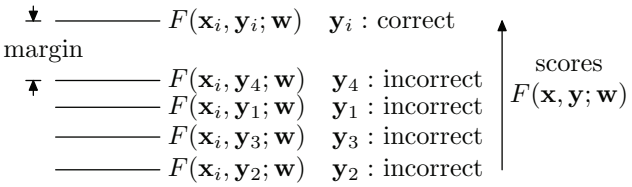

Fig. 1. The scoring function $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ in structured SVM. $\mathbf{y}_i$ is the correct structural label for the given feature vector $\mathbf{x}_i$; since there are many labels $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$, ..., etc, which are incorrect, we need to have the function F which gives the highest score $F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})$ for $\mathbf{y}_i$ and at the same time maximizes the margin between $F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})$ and the score $F(\mathbf{x}_i, \mathbf{y}_4; \mathbf{w})$ for the nearest incorrect label.

2.1.1. Margin Maximization

The parameter $\mathbf{w}$ mentioned above can be trained by structured SVM [6] similar to the ordinary SVM,

$$\min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i \quad \text{such that}$$
$$\forall i, \forall \mathbf{y} \in Y \backslash \mathbf{y}_i, \xi_i \geq 0 : \langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \qquad (3)$$

where $C > 0$ is a parameter that controls the trade-off between the training error minimization and margin maximization, ξ_i are the slack variables, and $\delta\Psi_i(\mathbf{y})$ is $\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y})$ for convenience. Besides, $\Delta(\mathbf{y}_i, \mathbf{y})$ is a loss function, which has the property $\Delta(\mathbf{y}_i, \mathbf{y}_i) = 0$, and $\Delta(\mathbf{y}_i, \mathbf{y}) > 0$ for $\mathbf{y} \neq \mathbf{y}_i$.

The above formulation is the primal form of the structured SVM. We can also employ the Lagrange multiplier $\alpha_{i\mathbf{y}}$ to enforce the margin constraint for each $(\mathbf{x}_i, \mathbf{y}_i)$, which then leads to the dual form of the above optimization problem,

$$\underset{\alpha}{\text{argmax}} \left[-\frac{1}{2}\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\langle\delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}})\rangle + \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\Delta(\mathbf{y}_i, \mathbf{y}) \right]$$
$$\text{such that} \sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} \leq \frac{C}{n}, \forall i = 1, \ldots, n. \qquad (4)$$

Both primal and dual form of this optimization problem can be approximated using the cutting-plane algorithm [7].

2.1.2. Sequence Decoding

Note that in the formulation here in fact $\mathbf{x}$ can be either a single feature vector or a sequence of feature vectors, and $\mathbf{y}$ can be either a single label or a sequence of labels. When $\mathbf{x}$ is a sequence of testing feature vectors, $\mathbf{x} = \{\mathbf{x}^t, t = 1, 2, \ldots, T\}$, where t is the time index, and we wish to decode the sequence into a sequence of labels, $\mathbf{y} = \{\mathbf{y}^t, t = 1, 2, \ldots, T\}$, this problem can be solved by dynamic programming, sometimes better known as Viterbi decoding. All possible values are kept during decoding, so that the maximization of $\langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y})\rangle$ is realized naturally.

2.2. Phoneme Recognition using Structured SVM

Here we consider how to map the phoneme recognition task to a structured SVM task as summarized above. One way to accomplish this is to treat it as a sequence labeling problem. We take the feature vectors of MFCC or PLP parameters for a sequence of signal frames as a sequence of feature vectors, here $\mathbf{x} = \{\mathbf{x}^t, t = 1, 2, \ldots T\}$, and the phoneme label for $\mathbf{x}^t$ is $\mathbf{y}^t$, where each phoneme corresponds to a different label. So the task is to decode $\mathbf{x} = \{\mathbf{x}^t, t = 1, 2, \ldots, T\}$ into the label sequence $\mathbf{y} = \{\mathbf{y}^t, t = 1, 2, \ldots, T\}$. Since the most successful and well known solution to this problem is HMM, we try to encode what HMM has been doing into the function $\Psi(\mathbf{x}, \mathbf{y})$ used here. An HMM consists of a series of states, and two most important sets of parameters — the transition probabilities between states, and the observation probability distribution for each state. Such a structure is slightly complicated for the structured SVM, so we use a simplified HMM with only one state for each phoneme. With this simplification, these two sets of probabilistic parameters can be estimated by adding up all the counts of the transition between labels (or states) and also adding up all the feature vectors for each label (phoneme or state). This is shown in Fig 2.

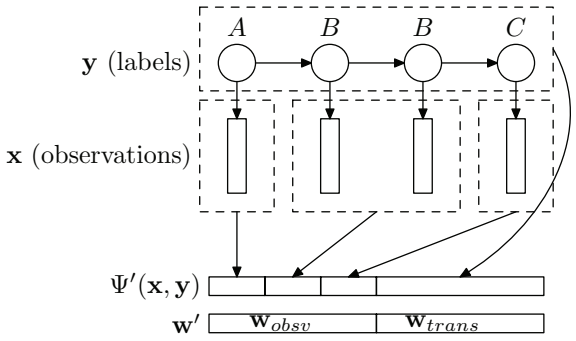

Fig. 2. A simplified example showing the definition of $\Psi(\mathbf{x}, \mathbf{y})$ and $\mathbf{w} = (\mathbf{w}'_{obsv}, \mathbf{w}'_{trans})'$, where A, B, C are labels (phonemes) and $\mathbf{x}$ are observation vectors.

We first define a vector helpful in the above purpose,

$$\Lambda(\mathbf{y}^t) = (\delta(L_1, \mathbf{y}^t), \delta(L_2, \mathbf{y}^t), \ldots, \delta(L_K, \mathbf{y}^t))' \in \{0, 1\}^K, \qquad (5)$$

where L_k's are possible labels (phonemes), K is the total number of different phonemes, and $\delta(L_k, \mathbf{y}^t)$ is 1 if $\mathbf{y}^t = L_k$ and 0 otherwise. So $\Lambda(\mathbf{y}^t)$ is a K-dimensional vector with its k-th component being 1 and all other components being 0 if $\mathbf{y}^t = L_k$. Tensor product $\otimes$ is handy here, which is defined as

$$\otimes : \mathbb{R}^P \times \mathbb{R}^Q \to \mathbb{R}^{PQ}, \quad (\mathbf{a} \otimes \mathbf{b})_{i+(j-1)P} \equiv a_i \times b_j, \qquad (6)$$

where $\mathbf{a}$ and $\mathbf{b}$ are two ordinary vectors, and P and Q are their dimentions respectively. The function $\Psi(\mathbf{x}, \mathbf{y})$ to be used in the scoring function $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y})\rangle$ can then be configured as the concatenation of two vectors,

$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \sum_{t=1}^{T} \mathbf{x}^t \otimes \Lambda(\mathbf{y}^t) \\ \sum_{t=1}^{T-1} \Lambda(\mathbf{y}^t) \otimes \Lambda(\mathbf{y}^{t+1}) \end{pmatrix}, \qquad (7)$$

where $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T)$ and $\mathbf{y} = (\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^T)$. The upper half of the right hand side of (7) is to accumulate the distribution of all components of $\mathbf{x}^t$ for each phoneme in the feature vector sequence $\mathbf{x}$, and locate them at different components of the vector $\Psi(\mathbf{x}, \mathbf{y})$i, as shown in the left half of the vector $\Psi'(\mathbf{x}, \mathbf{y})$. The lower half of the right hand side of (7), on the other hand, is to accumulate the transition counts between each pair of labels (states) in the label sequence $\mathbf{y}$, as shown in the right half of the vector

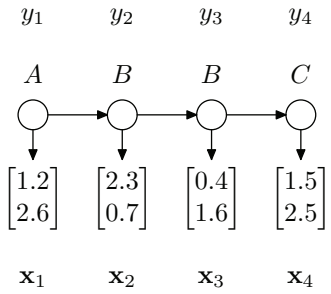

Fig. 3. A simplified example of feature sequence $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4)$ and label sequence $\mathbf{y} = (\mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3, \mathbf{y}^4) = (A, B, B, C)$

$\Psi'(\mathbf{x}, \mathbf{y})$. $\Psi(\mathbf{x}, \mathbf{y})$ is then the concatenation of the two, so it keeps the statistics of $\mathbf{x}^t$ for different phonemes for all $\mathbf{x}^t$ in $\mathbf{x}$, and the transitions between states for all $\mathbf{y}^t$ in $\mathbf{y}$. With enough training instances $(\mathbf{x}, \mathbf{y})$ and the function $\Psi(\mathbf{x}, \mathbf{y})$, we can then learn the scoring function $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ by training the parameter vector $\mathbf{w}$. Since $\Psi(\mathbf{x}, \mathbf{y})$ has two parts, the left for observation distributions and the right for transition probabilities, clearly the parameter vector $\mathbf{w}$ also has two parts, $\mathbf{w} = (\mathbf{w}'_{obsv}, \mathbf{w}'_{trans})'$, as shown in Fig 2. The vector $\Psi(\mathbf{x}, \mathbf{y})$ in (7) can be easily extended to higher order Markov assumptions (transition to the next state depending on more than one previous states). For example, by replacing the tensor term in upper half of (7) with $\sum_t \mathbf{x}^t \otimes \Lambda(\mathbf{y}^t) \otimes \Lambda(\mathbf{y}^{t+1})$ and the tensor term in lower half of (7) with $\sum_t \Lambda(\mathbf{y}^t) \otimes \Lambda(\mathbf{y}^{t+1}) \otimes \Lambda(\mathbf{y}^{t+2})$, we have the second order Markov assumption.

Consider a simplified example as shown in Fig 3. It is then easy to find that the upper part of $\Psi(\mathbf{x}, \mathbf{y})$ is $\sum_{t=1}^4 \mathbf{x}^t \otimes \Lambda(\mathbf{y}^t) = (1.2, 2.6, 2.7, 2.3, 1.5, 2.5)'$, and the lower half of $\Psi(\mathbf{x}, \mathbf{y})$ is $\sum_{t=1}^4 \Lambda(\mathbf{y}^t) \otimes \Lambda(\mathbf{y}^{t+1}) = (0, 0, 0, 1, 1, 0, 0, 1, 0)'$. We therefore have $\Psi(\mathbf{x}, \mathbf{y}) = (1.2, 2.6, 2.7, 2.3, 1.5, 2.5, 0, 0, 0, 1, 1, 0, 0, 1, 0)'$.

3. EXPERIMENTS

3.1. Experimental Setup and Baseline Results

The corpus we used in the experiments reported below is TIMIT, a collection of English read sentences. It consists of 6300 sentences, produced by 630 different speakers with eight different dialects. Each speaker read ten sentences — two dialect sentences, five phonetically compact sentences, and three phonetically diverse sentences. The whole corpus is divided into a training set and a testing set, with 462 speakers and 168 speakers each. It was reported that including the dialects can boost the performance so we discarded them [8]. The training set after discarding all the dialect sentences consists of 3696 sentences (462×8). Further more, there is a core testing set in the complete testing set with only 24 speakers and no dialect sentences, so it has 192 (24×8) sentences. We used the training set without dialect sentences for training and the core testing set for testing.

The TIMIT phoneme set originally has 64 phonemes, but in order to compare with results reported elsewhere [8], we employed the same settings as done earlier by removing all the glottal stops and merging some of the similar phonemes to have a phoneme set with 48 phonemes. Besides, there are seven groups where within-group confusions are not counted. Thus, the models were trained with a set of 48 phonemes and tested only with a set of 39 phonemes, which conformed to CMU/MIT standards [8].

The results with HMMs are listed in Table 1, used below as baselines to be compared with those with structured SVM. Two sets of phoneme HMMs were trained and used as baselines. Both were left-to-right with five states for each phoneme. We doubled the Gaussian mixtures every 20 EM iterations until the number of mixtures reaches 32. Because there are no exact time boundaries in TIMIT, we first trained an initial set of flat-start HMMs (left column of Table 1). After having the first set of HMMs, all the signal frames were forced aligned and used to train the second set of models (right column of Table 1). We used two different sets of initial plain features, the 39 MFCC parameters and 39 PLP parameters (the upper two rows of Table 1). To take a step further, we used Tandem with an MLP that has 9 frames or $39 \times 9 = 351$ parameters for input, 1000 for hidden layer, and 48 output posterior probabilities (the middle two rows of Table 1). We also used PCA to reduce the output parameters of the MLP from 48 to 37 (the lower two rows of Table 1). From the results in Table 1, we can find that the best performance of HMM with plain MFCC/PLP features was 63.26%, and the best for Tandem (with PCA) was 70.42%. These results are consistent with those reported previously [8].

Table 1. Phoneme accuracy in the baseline experiments using HMMs with different sets of features.

Features used in the HMMs		FS-HMM[a]	FA-HMM[b]
plain features	MFCC	62.47%	63.26%
	PLP	62.69%	62.91%
Tandem features	MLP-MFCC	68.77%	69.26%
	MLP-PLP	69.25%	69.50%
Tandem features with PCA	PCA-MLP-MFCC	70.19%	70.30%
	PCA-MLP-PLP	70.26%	70.42%

[a] HMM (flat start)
[b] HMM (after forced alignment)

3.2. Results with Structured SVM

For the structured SVM, we used exactly the same six sets of features as used in Table 1, i.e., the plain features, Tandem features, and those with PCA. The forced aligned training set used to train the second set of HMMs in Table 1 was used in training the structured SVM. We manually tuned the control constant C in (3). The results for exactly the same six sets of features as in Table 1 are listed in Table 2, which were obtained based on the dual form and the first order Markov assumption. Notice that in all cases the results were better if the control constant C was made larger to have better balance between error minimization and margin maximization. Also, it is not clear if the performance saturated at $C = 1000$. Better performance may be possible if a larger C can be used. However, the running time grows proportional to C. It became impractical to go for other larger C's. In addition, though both primal and dual form of structured SVM are of equal performance, dual form is slightly faster. Hence, all of the reported performances are based on dual forms of structured SVM.

By comparing Tables 1 and 2, we find that the performance of structured SVM is significantly lower than HMM if the plain MFCC/PLP features were used (top two rows of Tables 1 and 2). This is not surprising because in structured SVM here each phoneme has only one state, but in HMM five-state models were used. Also in HMM Gaussian mixture models were used, but there was no Gaussian modelling in structured SVM. In structured SVM we only added up the feature parameters which only gives something like a discrete

Table 2. Phoneme accuracy of the structured SVM for different sets of features with Dual form, first order Markov assumption and different values of the parameter C in (3).

Features used in structured SVM		$C = 1$	$C = 10$	$C = 100$	$C = 1000$
plain features	MFCC	38.87%	46.55%	49.74%	51.29%
	PLP	39.08%	44.81%	49.65%	51.22%
Tandem features	MLP-MFCC	57.40%	67.74%	70.86%	71.71%
	MLP-PLP	57.57%	67.65%	70.89%	71.75%
Tandem features with PCA	PCA-MLP-MFCC	56.71%	67.13%	70.18%	71.29%
	PCA-MLP-PLP	56.79%	67.20%	70.19%	71.32%

distribution.

However, it is very encouraging that with Tandem features (lower four rows in Tables 1 and 2) the performance of structured SVM jumped to another level, even better than HMM in all cases, although performing PCA seemed not helpful here. Note that here the best result for structured SVM is 71.75% with Tandem features, 1.33% higher than the best result for HMM (70.42%) with Tandem features with PCA, even if in structured SVM each phoneme has only one state and no Gaussian modelling was applied. This is probably because the structured SVM itself has better discriminative power. Also, the fact that Tandem features offered significantly better performance for structured SVM as compared to the plain features probably has to do with the missing of Gaussian modelling in structured SVM. The MLP may have produced a set of much better feature parameters of posterior probabilities for the structured SVM. Moreover, applying PCA didn't help for structured SVM (lower two rows compared to middle two rows in Table 2) seems to imply the structured SVM has better ability to extract useful information from relatively redundant data.

In order to measure the performance of structured SVM with HMM, we did an extra experiment on HMM, also with five states per phoneme but having different numbers of mixtures per state. We found the recognition accuracy of a five-state single Gaussian HMM is 50.33% roughly the same performance as a structured SVM with $C = 1000$. This may implies that with plain MFCC/PLP features the structured SVM has performance roughly corresponding to a five-state single Gaussian HMM.

We also tested the structured SVM for the second order Markov assumption. The initial results for $C = 1$ and 10 as listed in Table 3 are encouraging too. By comparing to Table 2, with Tandem features and second order Markov assumption implemented, the results for a value of smaller C is comparable to that with first order Markov assumption with a much larger C. Because the running time of the cutting-plane algorithm is linear time with respect to the training samples, and also runs in $O(C)$ if the number of training samples are fixed. This implies the use of higher order Markov assumption to achieve the same performance with smaller C is desirable.

4. CONCLUSION

In this paper we exploit the possibility of using structured SVM for phoneme recognition by configuring the basic framework of HMM into the structured SVM. Preliminary test results show that structured SVM offered much lower performance as compared to HMMs if plain MFCC/PLP features were used, probably because in the initial attempt here each phoneme has only one state even without Gaussian modelling. However with the help of MLP to produce posterior probabilities as Tandem features, it tied or even outperformed HMM. The generality, flexibility, efficiency, and performance of structured SVM may imply possible extensions and hopefully better performance beyond HMM in the future. Besides, other comparisons such as large margin HMM are underway. This is only an initial attempt. Much work is left for the future.

Table 3. Phoneme accuracy of structured SVM for different sets of features with dual form, second order Markov assumption and $C = 1$ and 10.

Features used in structured SVM		$C = 1$	$C = 10$
plain features	MFCC	39.03%	46.38%
	PLP	39.19%	44.61%
Tandem features	MLP-MFCC	64.43%	69.94%
	MLP-PLP	64.25%	70.07%
Tandem features with PCA	PCA-MLP-MFCC	63.65%	69.84%
	PCA-MLP-PLP	63.75%	69.91%

5. REFERENCE

[1] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[2] F. Sha and L. K. Saul, "Large margin hidden markov models for automatic speech recognition," *Advances in neural information processing systems*, 2007.

[3] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden markov support vector machines," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.*, 2003.

[4] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Proc. 18th International Conf. on Machine Learning*, pp. 282–289, 2001.

[5] J. Morris and E. Fosler-Lussier, "Discriminative phonetic recognition with conditional random fields," *HLT-NAACL workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, 2006.

[6] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.

[7] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Journal of Machine Learning*, 2009.

[8] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 11, pp. 1641–1648, Nov 1989.