ON THE USE OF SUPPORT VECTOR MACHINES FOR PHONETIC CLASSIFICATION

Philip Clarkson and Pedro J. Moreno*

Compaq Computer Corporation,
Cambridge Research Laboratory
One Kendall Square, Building 700
Cambridge, MA 02139
USA

ABSTRACT

Support Vector Machines (SVMs) represent a new approach to pattern classification which has recently attracted a great deal of interest in the machine learning community. Their appeal lies in their strong connection to the underlying statistical learning theory, in particular the theory of Structural Risk Minimization. SVMs have been shown to be particularly successful in fields such as image identification and face recognition; in many problems SVM classifiers have been shown to perform much better than other nonlinear classifiers such as artificial neural networks and k-nearest neighbors.

This paper explores the issues involved in applying SVMs to phonetic classification as a first step to speech recognition. We present results on several standard vowel and phonetic classification tasks and show better performance than Gaussian mixture classifiers. We also present an analysis of the difficulties we foresee in applying SVMs to continuous speech recognition problems.

1. INTRODUCTION

The theory of Support Vector Machines was first introduced by Vapnik and was developed from the theory of Structural Risk Minimization [14]. SVMs learn the boundary regions between samples belonging to two classes by mapping the input samples into a high dimensional space, and seeking a separating hyperplane in this space. The separating hyperplane is chosen in such a way as to maximize its distance from the closest training samples (a quantity referred to as the *margin*).

The appeal of SVMs is twofold. Firstly they do not need any fine tuning of parameters, and secondly they exhibit a great ability to generalize. In many problems SVMs have been shown to provide better performance than more traditional techniques, such as highly tuned neural networks. In recent years they have been used in multiple applications (see [1]), from vision problems to text classification. However, their application to speech recognition problems has been very limited. In this paper we focus on the simpler task of phonetic classification. Studying the performance of SVMs in phonetic classification tasks will allow us to understand the issues involved in the more difficult task of speech recognition.

This paper is organized as follows: In section 2 we give a brief introduction to the theory of SVMs. In section 3 we describe experiments using SVMs for vowel classification using the Peterson and Barney and the Deterding data sets. In section 4 we describe experiments in phonetic classification using the more difficult and general phonetically balanced TIMIT database. We conclude with a discussion of the remaining issues which need to be addressed in order to make SVMs useful for continuous speech recognition.

*Philip Clarkson is a Ph.D. student at Cambridge University Engineering Department, United Kingdom. This work was conducted during a summer internship at Cambridge Research Laboratory, Massachusetts.

2. SUPPORT VECTOR MACHINES

This section introduces the theory behind SVMs. Lack of space prohibits a more detailed discussion, but interested readers are referred to [14] for an in depth discussion or to [1] for a short tutorial.

2.1. The Linearly Separable Case

Suppose we have a set of training samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ where $\mathbf{x}_i \in \mathbb{R}^n$. Each sample has a corresponding label $y_1, y_2, \ldots, y_m$ (where $y_i \in \{-1, 1\}$) that indicates which of two classes each sample belongs to. Then the hyperplane $(\mathbf{w} \cdot \mathbf{x}) + b$ separates the data if and only if

$$(\mathbf{w} \cdot \mathbf{x}_i) + b > 0 \quad \text{if} \quad y_i = 1 \tag{1}$$
$$(\mathbf{w} \cdot \mathbf{x}_i) + b < 0 \quad \text{if} \quad y_i = -1 \tag{2}$$

and we can scale $\mathbf{w}$ and b so that this is equivalent to

$$(\mathbf{w} \cdot \mathbf{x}_i) + b \geq 1 \quad \text{if} \quad y_i = 1 \tag{3}$$
$$(\mathbf{w} \cdot \mathbf{x}_i) + b \leq -1 \quad \text{if} \quad y_i = -1 \tag{4}$$

or

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 \quad \forall \quad i \tag{5}$$

To find the optimal separating hyperplane, we need to find the plane which maximizes the distance between the hyperplane and the closest sample. The distance of the closest sample is

$$d(\mathbf{w}, b) = \min_{\{\mathbf{x}_i | y_i = 1\}} \frac{\mathbf{w} \cdot \mathbf{x}_i + b}{|\mathbf{w}|} - \max_{\{\mathbf{x}_i | y_i = -1\}} \frac{\mathbf{w} \cdot \mathbf{x}_i + b}{|\mathbf{w}|} \tag{6}$$

and from equation (4) we can see that the appropriate minimum and maximum values are ± 1. So we need to maximize

$$d(\mathbf{w}, b) = \frac{1}{|\mathbf{w}|} - \frac{-1}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|} \tag{7}$$

Therefore our problem is equivalent to minimizing $|\mathbf{w}|^2/2$ subject to the constraints expressed in (5). By forming the Lagrangian, and solving the dual problem, this can be translated into the following [1]:

Minimize

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{8}$$

subject to

$$\begin{aligned} \alpha_i &\geq 0 && (9)\\ \sum_i \alpha_i y_i &= 0 && (10) \end{aligned}$$

The α_i are the Lagrange multipliers; there is one Lagrange multiplier for each training sample. The training samples for which the Lagrange multiplier is non-zero are called *support vectors*, and are such that the equality in equation (5) holds. The samples with Lagrange multipliers of zero could be removed from the training set without affecting the position of the final hyperplane.

This is a well understood quadratic programming problem, and software packages exist which can find a solution. Such solvers are non-trivial, however, especially in cases where we have large training sets [9].

2.2. The non-separable case

The optimization problem described in the previous section will have no solution if the data is not separable. In order to cope with this scenario, we modify the constraints (1) and (2) such that the constraints are looser, but a penalty is incurred for misclassification:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b &\geq 1 - \xi_i \text{ if } y_i = 1 && (11)\\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq \xi_i - 1 \text{ if } y_i = -1 && (12)\\ \xi_i &\geq 0 \quad \forall \quad i && (13) \end{aligned}$$

If $\mathbf{x}_i$ is to be misclassified, we must have $\xi_i > 1$, and hence the number of errors is less than $\sum_i \xi_i$. So we may add a penalty for misclassifying training samples by replacing the function to be minimized by $|\mathbf{w}|^2/2 + C(\sum_i \xi_i)$, where C is a parameter which allows us to specify how strictly we want the classifier to fit to the training data.

If we form the Lagrangian, the dual problem now becomes:

Minimize

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{14}$$

subject to

$$\begin{aligned} 0 \leq \alpha_i \leq C && (15)\\ \sum_i \alpha_i y_i = 0 && (16) \end{aligned}$$

2.3. The non-linear case

The classification framework outlined above is limited to linear separating hyperplanes. SVMs get around this problem by mapping the sample points into a higher dimensional space using a non-linear mapping chosen in advance. That is, we choose a map $\Phi : \mathbb{R}^n \mapsto \mathcal{H}$ where the dimension of $\mathcal{H}$ is greater than n. We then seek a separating hyperplane in the higher dimensional space, this is equivalent to a non-linear separating surface in $\mathbb{R}^n$.

The data only ever appears in our training problem (equations (8) to (10)) in the form of dot products, so in the higher dimensional space we are only dealing with the data in the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. If the dimensionality of $\mathcal{H}$ is very large, then this could be difficult, or very computationally expensive to compute. However, if we have a *kernel function* such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, then we can use this in place of $\mathbf{x}_i \cdot \mathbf{x}_j$ everywhere in the optimization problem, and never need to know explicitly what Φ is.

Some examples of kernel functions are the polynomial kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$ and the Gaussian radial basis function (RBF) kernel $K(\mathbf{x}, \mathbf{y}) = e^{|\mathbf{x}-\mathbf{y}|^2/2\sigma^2}$.

2.4. Multi-class classifiers

So far we have only discussed using SVMs to solve two-class problems. However, if we are interested in conducting phone classification experiments, we will need to choose between multiple classes. The best method of extending the two-class classifiers to multi-class problems is not clear. Previous work has generally constructed a "one vs. all" classifier for each class [12], or constructed a "one vs. one" classifier for each pair of classes. The "one vs. all" approach works by constructing for each class a classifier which separates that class from the remainder of the data. A given test example $\mathbf{x}$ is then classified as belonging to the class whose boundary maximizes $(\mathbf{w} \cdot \mathbf{x}) + b$. The "one vs. one" approach simply constructs for each pair of classes a classifier which separates those classes. A test example is then classified by all of the classifiers, and is said to belong to the class with the largest number of positive outputs from these sub-classifiers.

In [15] a method of extending the quadratic programming problem to multi-class problems is presented, although the results presented suggest that it performs no better than the more ad-hoc methods of building multi-class classifiers from sets of two-class classifiers.

3. VOWEL CLASSIFICATION

Our preliminary experiments focus on vowel classification tasks, based on the Deterding [11] and Peterson and Barney [10] data sets. There are two advantages of starting with vowel classification tasks. Firstly the problem is small – the data is low dimensional, there are only ten classes to choose between, and there are a relatively small number of training samples. Secondly, vowels do not vary much in time, and we can therefore characterize them easily with a vector of fixed length. For example, the vowels are characterized by their four formant frequencies in the Peterson and Barney data, and by ten LPC reflection coefficients in the Deterding data set.

To provide a comparison point with SVMs we provide results of experiments performed with mixtures of Gaussians. The Gaussians were initialized with the k-means algorithm which provided a starting point for the well-known expectation maximization (EM) algorithm [2]. The EM algorithm was trained until convergence of the log likelihood was achieved. The covariances in the Gaussians were modeled with diagonal matrices. Our baseline classification results with Gaussian mixture models were 62.1% accuracy for the Deterding data using 16 Gaussians per class, and 81.7% accuracy for the Peterson and Barney set using 3 Gaussians per class.

Table 1 presents our results using SVMs on the vowel classification tasks, using both a polynomial kernel of degree 4 and an RBF kernel. We have investigated constructing multi-class classifiers using a one vs. all approach and a one vs. one approach.

Data Set	**Kernel**	**Multi-class**	**Accuracy**
Deterding	Polynomial	one vs. all	54.9%
Deterding	Polynomial	one vs. one	60.4%
Deterding	RBF	one vs. all	66.1%
Deterding	RBF	one vs. one	70.0%
PB	Polynomial	one vs. all	77.9%
PB	Polynomial	one vs. one	84.5%
PB	RBF	one vs. all	79.6%
PB	RBF	one vs. one	85.3%

Table 1: Vowel classification results

The results are extremely encouraging. Not only do the SVM classifiers perform comparably with the Gaussian classifiers (and better in many cases), but they perform significantly better than previously reported results using neural network classifiers for the Deterding data [11][1]. Furthermore, our results on the Peterson and Barney data compare favorably to previously reported results [7, 13].

Other conclusions which can be drawn from these results are that the RBF kernel outperforms the polynomial kernel, and that constructing multi-class classifiers from a set of one vs. one classifiers yields better performance than using a set of one vs. all classifiers.

4. TIMIT EXPERIMENTS

To test the performance of SVMs on a more difficult task we used the TIMIT database [3]. Training was performed on the 'sx' and 'si' training sentences. These create a training set with 3696 utterances from 168 different speakers. For testing we chose the core set. It consists of 192 utterances from 24 different speakers not included in the training set. All utterances contain labels indicating the phone identity and the starting and ending time of each phone. The standard Kai-Fu Lee clustering [8] was used, resulting in a set of 39 phones.

All of our experiments on the TIMIT database were conducted by constructing multi-class classifiers using one vs. one classifiers. This was principally motivated by the fact that they had been shown to perform better than one vs. all classifiers on the vowel classification tasks. There were, however, other practical issues for this choice – using one vs. one classifiers makes each individual training problem smaller, and hence the memory and CPU time required to train each classifier is greatly reduced. While using a one vs. all approach requires many fewer classifiers to be trained, the memory requirements to train each classifier were found to be prohibitive.

The key problem with conducting classification experiments with the TIMIT database is that the segments that we are seeking to classify are not of a uniform length. In order to use the SVM classifiers, however, we must encode the waveform information in a fixed-length vector. Furthermore, unlike the case of vowel classification, it is not sufficient to take a sample at one point in the example, as phones other than vowels can vary significantly in time, and it is often these time-varying dynamic patterns that are critical in performing classification.

We chose a simple method of encoding the variable length segment information in a vector of fixed length. We converted the utterances from their waveform representation into a sequence of 13 dimensional mel cepstral feature vectors, their time derivatives and second order derivatives. The cepstra and its time derivatives were combined into a 39 dimensional vector. For our cepstral analysis we used a 25.5 ms Hamming window shifted every 10 ms. Each phone segment was broken into three regions in the ratio 3-4-3. The 39 dimensional vectors belonging to each of these regions were averaged resulting in three 39 dimensional vectors[2]. In addition, the 39 dimensional vectors belonging to a window region centered at the start of the phonetic segments and with a 40 ms width were averaged, resulting in another 39 dimensional vector. The same was done for a window centered at the end of the segment. One additional feature indicating the log-duration of the phone segment was added. This resulted in a vector with 196 components.

The results of using SVM classifiers with various kernel functions to perform phonetic classification are shown in Table 2. In order to provide a comparison point we conducted experiments using Gaussian mixture models. The initialization and training algorithms used to learn the parameters of the Gaussian mixtures were the same as those used in the vowel classification experiments. Using 64 diagonal covariance Gaussians per class we obtained a classification accuracy of 73.7%.

Kernel Function	**SVM Accuracy**
Polynomial degree 3	76.4%
Polynomial degree 4	77.1%
Polynomial degree 5	77.6%
Polynomial degree 6	77.0%
Radial Basis Function	76.3%

Table 2: TIMIT phonetic classification results

These results show that SVMs perform significantly better than the Gaussian classifiers. Furthermore, the results are competitive with current state-of-the-art performance in phonetic classification using this data set [5, 16]. It is also interesting to note that the choice of kernel function does not have a major impact on accuracy.

5. CONTINUOUS SPEECH RECOGNITION

The results given in the previous sections show that support vector machines can perform well on phonetic classification tasks. However, a number of obstacles remain before they can be useful in the context of continuous speech recognition. This section explores these issues, and discusses how they might be resolved.

5.1. Context-dependent models

It is well established that many phones vary greatly when they occur in different phonetic contexts. Current speech recognition

[1] Our results here are consistent with those presented in [4]

[2] This mirrors the acoustic representation reported in [5].

systems based on hidden Markov models handle this variability by creating a separate model for all sufficiently different phonetic contexts (where by "phonetic context" we mean the phones which occur immediately before and after the current phone). These are referred to as *triphone* models. This obviously has the potential to increase the number of possible models required by a power of three.

At present, our one vs. one classifier requires us to train $O(n^2)$ classifiers, where n is the number of classes. If we increase the number of "classes" to n^3, we would therefore require $O(n^6)$ classifiers to be trained, which is clearly out of the question. The standard engineering solutions used in large vocabulary speech recognition systems, such as clustering of triphones and the use of a tree based hierarchy of classifiers could possibly alleviate the problem.

5.2. Estimation of *a posteriori* probabilities

A more critical problem than the practical issue of the number of classifiers needed is what the classifiers actually tell us. Speech recognition takes place on a probabilistic basis – the identity of the most likely phone is not sufficient information, we would like a probability distribution over all phones. It is unclear how to adapt our approach in order to estimate these probabilities, although it is possible that one could generate a probability estimate based on the distance of a test sample from each of the separating hyperplanes.

5.3. Modeling of time-varying dynamics

A further drawback with using SVMs for speech recognition is that as described here they are "static" models; they are incapable of successfully modeling the time-varying dynamics of the speech signal in the way that hidden Markov models or recurrent neural networks [11] are. However, work on the use of SVMs to handle patterns of variable length [6] suggests that they could be used successfully to model the dynamics of the speech signal.

6. CONCLUSIONS AND FUTURE WORK

These experiments have demonstrated that SVMs can be applied successfully to phonetic classification. We have shown that they outperform classifiers based on mixtures of Gaussians for all of the data sets. The vowel classification rates which we have observed using SVMs compare favorably with those reported in the literature using neural networks. Furthermore, the classification rates on the 39 phone TIMIT core set are comparable to state of the art performance for this task. However, there remains a lot of ground to be covered before SVMs can be successfully applied to continuous speech recognition. Several difficult problems still remain to be solved. This paper represents a preliminary step in understanding the problems of applying SVMs to speech recognition.

7. ACKNOWLEDGMENTS

The authors wish to thank Edgar Osuna and Tomaso Poggio from the Center for Biological and Computational Learning at MIT for providing the core Support Vector Machine software upon which the experiments were based, and for several helpful discussions. We also thank Andrew Hallberstadt and James Glass from the MIT Spoken Language Systems group for providing us with data.

8. REFERENCES

[1] Christopher Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2), 1998.

[2] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data Using the EM Algorithm. *Journal of the Royal Society of Statistics*, 39(1):1–38, 1977.

[3] W. Fisher, G. Doddington, and K. Goudie-Marshall. The DARPA Speech Recognition Research Database: Specifications and Status. In *Proceedings DARPA Speech Recognition Workshop*, pages 93–99, 1986.

[4] Aravind Ganapathiraju, Jonathan Hamaker, and Joseph Picone. Support Vector Machines for Speech Recognition. In *Proceedings ICSLP*, Sydney, Australia, 1998.

[5] Andrew Hallberstadt and James Glass. Heterogeneous Acoustic Measurements for Phonetic Classification. In *Proceedings Eurospeech*, Rhodes, 1997.

[6] Tommi S. Jaakola and David Haussler. Expoloting generative models in discriminating classifiers. In *Proceedings NIPS*98*, 1998.

[7] Visakan Kadirkamanthan and Mahesan Niranjan. Application of an Architecturally Dynamic Network for Speech Pattern Classification. *Proceedings of the Institute of Acoustics*, 14(6):343–350, 1992.

[8] K.-F. Lee and H. Hon. Speaker-Independent Phone Recognition Using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11), 1989.

[9] Edgar Osuna. *Support Vector Machines: Training and Applications*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1998.

[10] G.E. Peterson and H.L. Barney. Control Methods Used in a Study of the Vowels. *JASA*, 24:175–184, 1952.

[11] Tony Robinson. *Dynamic Error Propogation Networks*. PhD thesis, Cambridge University Engineering Department, February 1989.

[12] Berhard Scholköpf, Chris Burges, and Vladimir Vapnik. Extracting Support Data for a Given Task. In *Proceedings, First International Conference on Knowledge Discovery and Data Mining*, 1995.

[13] R.S. Shadafan and M. Niranjan. A Dynamic Neural Network Architecture by Sequential Partitioning of the Input Space. Technical report, Cambridge University Engineering Department, May 1995.

[14] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[15] J. Weston and C. Watkins. Multi-class Support Vector Machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May 1998.

[16] Stephen Zahorian, Peter Silsbee, and Xihong Wang. Phone Classification with Segmental Features and a Binary-Pair Partitioned Neural Network Classifier. In *Proceedings IEEE ICASSP*, 1997.