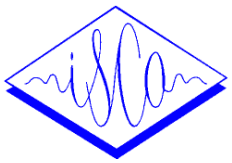

6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

Using Support Vector Machines for Spoken Digit Recognition

Issam Bazzi and Dina Katabi

MIT Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge Massachusetts 02139, USA
mailto: {issam,dinaktbi}@mit.edu

ABSTRACT

Recently Support Vector Machines (SVM) have emerged as a pattern classifier that can deal with a large feature space and a small data set. In this paper, we look at using SVMs for automatic speech recognition. We focus on a fairly simple speech recognition task: recognizing isolated spoken digits in English. The approach relies solely on the acoustic signal and does not utilize any phonological rules or language constraints in the recognition process. For each digit, a 420-dimension feature vector is extracted. The feature vector is derived from the Mel-Frequency cepstral coefficients of the speech signal. The 420 features are then reduced to a smaller number using principal component analysis (PCA). To perform N-way classification for the ten digits using the standard 2-class SVM classifiers, we examine scoring and voting classification schemes. The best performance is obtained with an N-way 1-versus-9 SVM classifier with a Gaussian kernel of a variance of 4 and using the first 45 PCA features. The accuracy of this classifier is 94.9%.

1. INTRODUCTION

Support Vector Machines (SVM) is a novel pattern classification technique that has been successfully applied to many pattern recognition problems in the areas of machine vision, text classification, marketing, and medicine [4]. The SVM framework is based on minimizing the expected risk of making a classification error. In contrast to traditional Bayesian classification methods, where probability density estimation is required for each class, an SVM does not require any density estimation. An SVM transfers the training data to a high dimensional space via a kernel function. Then, it finds an optimal separating hyper-plane that divides this high dimensional space into decision regions. The appeal of the SVM framework for speech recognition is twofold. First, SVMs have a great ability to generalize even with a small number of examples and a high-dimension feature space. Second, a trained SVM needs a small amount of computation to perform the recognition task. As a result, SVM constitutes an inexpensive classifier for systems with a small vocabulary.

In this paper, we address the problem of recognizing isolated spoken digits in English. Although the problem is simple, it allows us to examine the potential use of SVMs for speech recognition, the preprocessing required to extract a fixed size feature vector from the temporal speech signal, and the size of the feature space that can be used in this recognition framework. We also explore methods to extend the 2-class SVM classifier to perform multi-class classification.

In section 2, we give an overview of SVMs. Section 3 describes the feature extraction process. In section 4, we present our approach for performing recognition with SVMs. Sections 5 and 6 describe the experiments and the obtained results.

2. SUPPORT VECTOR MACHINES

This section provides a brief overview of Support Vector Machines. For a complete discussion, please refer to [1] or [4].

2.1 Structural Risk Minimization

SVM is based on the theory of Structural Risk Minimization (SRM). The 2-class classification problem addressed by this framework is formulated as follows. Given a set of ℓ labeled examples:

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell), x_i \in \Re^N, y_i \in \{1, -1\}$$

and a class of functions:

$$S = \{ f_a : a \text{ is an index} \}, f_a : \Re^N \rightarrow \{1, -1\},$$

we want to find the function in S that minimizes some error function. The straightforward approach uses the empirical risk as the error function to be minimized:

$$R_{emp}(a) = \frac{1}{\ell} \sum_{i=1}^{\ell} | f_a(\mathbf{x}_i) - y_i |$$

Minimizing the empirical risk minimizes the error on the training data but usually does not generalize well. Ideally for good generalization, the solution function (the classifier) should minimize the expected risk defined as follows.

$$R(a) = \int | f_a(\mathbf{x}_i) - y_i | \, dP(\mathbf{x}, y) \cdot$$

The problem however with minimizing the expected risk is that *P(**x**,y)*, which is the probability distribution according to which the data is drawn, is unknown. The SRM principle [1] addresses this problem by minimizing the following upper bound on the expected risk.

$$R(a) \leq R_{emp}(a) + f\left(\frac{h}{\ell}, \frac{\log(h)}{\ell}\right)$$

Where:

$$f(\frac{h}{\ell}, \frac{\log(h)}{\ell}) = \sqrt{\frac{h(\log\frac{2\ell}{h}+1) - \log(h/4)}{\ell}}$$

The parameter f is called the VC-confidence, while the parameter h is called the VC-dimension of the class of functions and is a measure of its complexity. This bound shows that to control the expected risk we need to control both R_{emp} and h. The empirical risk (R_{emp}) depends on the choice of the function and can be controlled by picking the right f_a in S. The VC-dimension (h) depends on the complexity of the class of functions S and its ability to generate functions that can separate the data for any possible labeling. To control h, a structure of nested classes of functions $S_1 \subset S_2 \subset \ldots \subset S_n$ such that $h_1 \leq h_2 \leq \ldots \leq h_n$ is used. By gradually increasing the complexity of the class of functions, This hierarchy can be searched for the class S_j that minimizes the above bound on the expected risk.

2.2 Linear Support Vector Machines

SVMs result from the application of the SRM principle to the case where the class of functions is the class of linear hyper-planes in the space of the data. If the data points are $\mathbf{x}_1,\ldots,\mathbf{x}_\ell \in \Re^N$, then a unique hyper-plane $\{\mathbf{x} \in \Re^N : (\mathbf{w}.\mathbf{x}) + b = 0\}$ in $\Re^N$ corresponds to the pair $(\mathbf{w}, b)$, if we additionally require $\min_{i=1,\ldots\ell} |\mathbf{w}.\mathbf{x}_i + b| = 1$. It is possible to prove that for the case of linear hyper-plane, minimizing the VC-confidence (f) can be reduced to minimizing the dot product ($\mathbf{w}.\mathbf{w}$). For lack of space, we do not provide a proof and refer the interested reader to [1].

If the data is linearly separable then the empirical risk can be reduced to zero. Given the above definition of a unique hyper-plane, we can force the solution hyper-plane to correctly classify all the training examples and have an empirical risk of zero by imposing the set of constraints $y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1$. Now we need to find the hyper-plane that satisfies the above constraints and has the minimal VC-confidence. Thus, the problem is to minimize ($\mathbf{w}.\mathbf{w}$) subject to $y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1$. By forming the Lagrangian and solving the dual problem, we find that the above optimization problem can be transformed to:

$$\textit{Maximize}: \sum_i a_i - \tfrac{1}{2}\sum_{i,j} a_i a_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$
$$\textit{Subject to}: a_i \geq 0 \ \textit{and} \ \sum_i a_i y_i = 0$$

and that the solution hyper-plane satisfies $\mathbf{w} = \sum y_i a_i \mathbf{x}_i$ and $a_i(y_i(\mathbf{w}.\mathbf{x}_i + b) = 0$, where α_i's are the Lagrange multipliers. This is the well-understood quadratic-programming problem [4]. The training points for which the Lagrange multipliers are greater than zero are called the support vectors, and they uniquely specify the solution. On the other hand, training points with zero Lagrange multipliers do not contribute to the solution.

When the data is not linearly separable, a hyper-plane cannot classify all data points correctly, and the empirical risk can't be reduced to zero. This case is handled by introducing slack variables to penalize for points on the wrong side of the separating hyper-plane.

2.3 Non-Linear Extensions: Kernels

Non-linear decision boundaries can be obtained by mapping the data points into a higher-dimension space using non-linear mapping functions, then applying the linear techniques in the new space. The fact that the data appears in the quadratic-programming problem only in the form of dot products means that we don't need to know the exact form of the mapping functions, we only need to know their dot products. Thus, a non-linear mapping of the data can be done by directly replacing the dot product $(\mathbf{x_j}.\mathbf{x_i})$ by certain non-linear kernel function $k(\mathbf{x_j},\mathbf{x_i})$ (e.g.: Gaussian or polynomial) allowing for a non-linear extension of SVMs.

3. FEATURE EXTRACTION

The input to the SVM classifier is a fixed-size feature vector that represents a whole word (digit). The SVM feature vector is generated from the Mel Frequency cepstral coefficients (MFCCs). However, concatenating all MFCC vectors associated with a word into one SVM feature vector would result in a variable size feature set since different words have different duration. The approach we adopted is based on removing MFCC vectors at points in time where the MFCCs change the least until only n MFCC vectors are left in the feature set, for some fixed n. For example if a word has N MFCC vectors, the following algorithm is used:

For N-n steps:

1. *Compute the distances $d_i = d(MFCC_i, MFCC_{i+1})$ for all remaining MFCC vectors*
2. *Find i for which d_i is minimum*
3. *Remove vector $MFCC_i$ from the feature set*

Where $d(MFCC_i, MFCC_{i+1})$ is a distance metric between two MFCC vectors $MFCC_i$ and $MFCC_{i+1}$. Applying this algorithm leaves us with n MFCC vectors per digit which when concatenated generate a fixed-size SVM feature vector. We experimented with few distance measures including Euclidean and absolute. For our work, we used n=30 vectors with MFCC vectors of size 14 resulting in 420 (30x14) dimensions. PCA is then used to reduce the feature set further. Figure 1 shows the first MFCC feature vector before and after applying the algorithm above to a sample of the digit "three". We note that the down-sampled version preserves most of the details seen in the original signal.

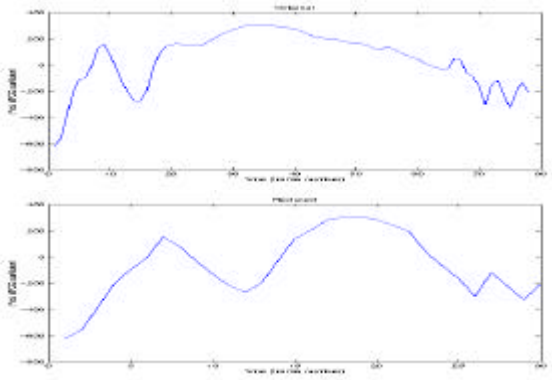

Figure 1: 1st MFCC vector and its reduced version.

4. THE RECOGNITION FRAMEWORK

An SVM is only a binary or a 2-way classifier. For our problem of digit recognition, we need to be able to classify among 10 classes. Performing N-way classification with a 2-way classifier is usually done using either a voting or a scoring mechanism that combines a set of binary classifiers. In the following two sections, we present two approaches for performing 10-way classification with an SVM.

4.1 1-Versus-N SVM Classifiers

Our first approach was to train 10 binary classifiers. Each of the binary classifiers is trained to recognize a particular digit from all other digits, so the training data of such a binary classifier is divided into positive examples, which correspond to the digit the classifier recognizes, and negative examples, which correspond to all of the other 9 digits. To obtain roughly the same amount of training data for the positive and negative classes, we weighed the data from the digit recognized by the classifier by a factor of 9.

Having these 10 classifiers, we can test membership of a test point by running it through all of them. If all but one test negative, the one that tests positive will be our classification choice. This is not usually the case since more than one classifier could test positive (or none). Hence, we used a more general scheme: the class of choice is C_{opt} such that:

$$C_{opt} = \text{argmax}_i \; d_{SVM}[C_i]$$

Where $d_{SVM}[C_i]$ is the distance of the test point from the separating hyper-plane for classifier C_i. The idea behind this approach can be explained as follows. If more than one class test positive, the one for which the data point has the largest distance from the separating hyper-plane is the one with the highest confidence. On the other hand, if none of the classifiers tests positive, the one with the largest distance (all distances are negative here) is the one where the data point is closest to the separating hyper-plane and is most likely to be mis-classified.

4.2 Pair-Wise SVM Classifiers

In the second approach, we build classifiers for each pair of digits, hence the term pair-wise classifiers. For 10 digits, the number of classifier is 45 (10-choose-2). Running the test point through each classifier gives us 45 hypothesis for the possible class the point could belong to. The class of choice is C_{opt} such that:

$$C_{opt} = \text{argmax}_i \; N_{SVM}[C_i]$$

Deciding on the class is done via a voting scheme among all classifiers. That is we add up the number of times a particular digit is chosen and we pick the one with the largest number $N_{SVM}[C_i]$. We break the tie between two or more digits by considering the distance from the separating hyper-plane as we did in the 1-versus-N approach.

5. EXPERIMENTS

For our work we used the 1992 and 1994 OGI speech database (collected at the Oregon Graduate Institute). For the experiments we conducted, we used the digit utterances from 133 speakers. Each speaker was asked to record several utterances saying the digits "one" through "ten". Using the MIT SUMMIT speech system [3], we segmented the speech into isolated digits to use with our SVM classifiers.

We divided the data set into two thirds for training and one third for testing. We used a total of 826 digits for training and 454 digits for testing. These numbers were for all the digits "one" through "nine", so there was around 83 training data points per digit and 45 testing data points per digit.

6. RESULTS

For all our experiments, we measured the performance as the fraction of the test points that were correctly classified. For the baseline, we used a Gaussian mixture classifier, where the best accuracy was 90.7% (9.3% classification error rate). In the following subsections, we report on a series of experiments that involve varying different parameters of the SVM classifier.

6.1 Effect of the Degree of the Polynomial Kernel

In the first set of experiments, we investigate the effect of the degree of the polynomial kernel. We consider a linear kernel to be a polynomial whose degree is 1. We ran our experiments for both the 1-versus-9 classifier and the pair-wise classifier described in section 4. We use a feature vector that represents the first 60 principal components of the data.

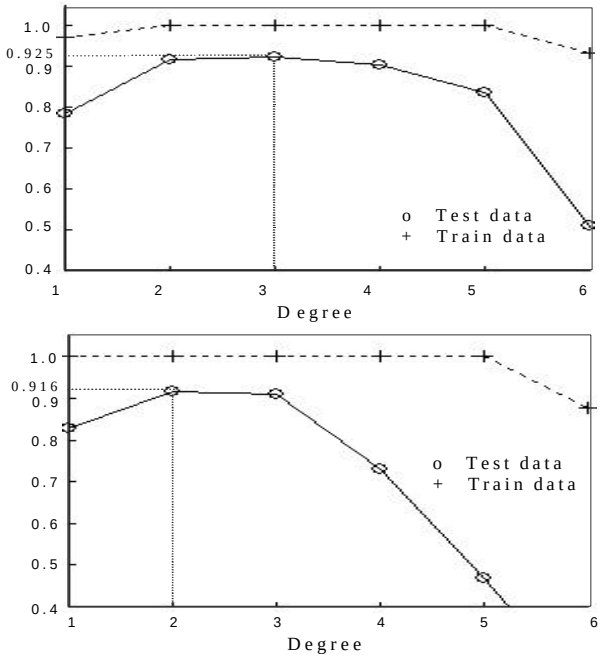

Figure 2: The accuracy of the polynomial kernel as a function of its degree. Top: 1-versus-N. Bottom: pair-wise

Figure 2 shows the accuracy of the classification as a function of the degree of the kernel for both the training and the test data. The graphs indicate that a kernel whose degree is 2 or 3 has the best performance. They also indicate that the best accuracy

obtained with a polynomial kernel for both classifiers is around 92%. In addition, Figure 2 shows that the N-way 1-versus-N classifier (top figure) works over a wider range of polynomial kernels, while the N-way pair-wise classifier degrades sharply for large degree polynomials.

6.2 Effect of the Variance in the Gaussian Kernel

In this section, we explore the effect of varying the variance of the Gaussian kernel. Results are shown in Figure 3 for the 1-versus-N classifier with 60 PCA features. The pair-wise classifier has similar performance, but was omitted here due to lack of space.

Figure 3 shows that a Gaussian kernel exhibits a better performance than a polynomial kernel. The best performance for the 1-versus-N (and pair-wise) classifier is 94%. It is achieved for a variance of 6. For lower values of the variance, the function is too flexible given the limited amount of training data available. For larger variances, the kernel function is too restricted for the problem at hand.

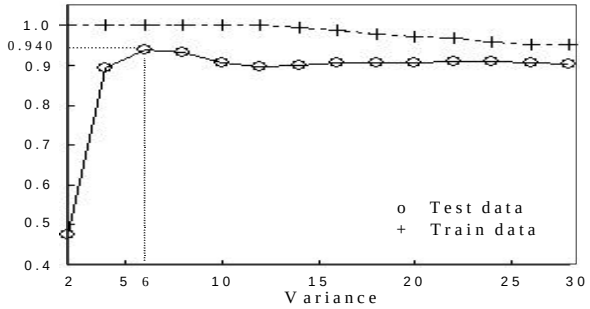

Figure 3: 1-vs-N classifier performance vs Gaussian variance.

6.3 Effect of the number of PCA features

Figure 4 illustrates the effect of changing the number of PCA features on the accuracy of the 1-versus-N classifier with a Gaussian kernel. The graphs indicate that we obtain the best performance when we use 45 PCA features. If we use less features (for example, 30 features) we leave out some essential information and the accuracy of the classifier degrades. If we use more features (90 features), then the number of features becomes too large given the available training points. The second point to notice in Figure 4 is that for a small number of features (30 and 45 features), the variance of the best Gaussian classifier is 4. Yet, as the number of features increases the variance of the best gaussian classifier increases as well (best variance is 6 for 60 and 90 features). This behaviour is expected since the effect of a large variance is somehow similar to that of a small number of PCA features; they both limit the flexibility of the resulting separating decision boundary.

7. DISCUSSION

Our best performing SVM classifier achieves a 94.9% accuracy on digit recognition and outperforms a Gaussian mixture classifier for the same training and testing data. However, current speech recognition systems can achieve 99% or more accuracy for the task of spoken digit recognition for phone numbers [2]. Given the scope of this work, the simplified approach we followed, and the fact that our method was purely based on the acoustic evidence, we believe that our results are quite encouraging as our first attempt to using the SVM framework for speech recognition. Current speech systems rely heavily on various sources of information that we did not utilize in our approach.

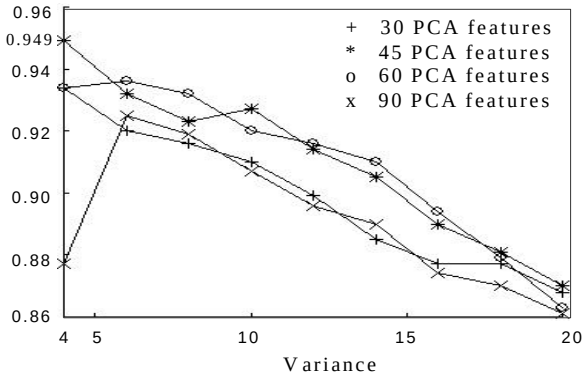

Figure 4: The Effect of varying the number of PCA features

First, the approach we followed does not model any of the spoken digits phonetic units. All digits are treated to have the same size and to be directly comparable via a large fixed size feature vector. Second, other sources of information that speech systems use were not taken advantage of. For example, using a language model (e.g. a digit n-gram) can improve performance. Finally, the amount of training data can be a key issue. In our case, we trained the system on only 826 data points for all digits. However for reported results in the literature, the amount of training data is on the order of 20,000 to 30,000 data points.

8. CONCLUSION

In this paper, we presented an approach for using SVMs for the simple speech recognition task of recognizing isolated spoken digits. The best performance was 94.9% accuracy for a 1-versus-9 classifier with a Gaussian kernel of a variance of 4 and using the first 45 PCA features. There is much that can be done to improve the performance such as the use of more training data and incorporating phonological and pronunciation rules and higher-level language constraints into the recognition process.

Acknowledgment: The authors would like to thank Thomaso Poggio and James Glass for their feedback and comments.

9. REFERENCES

[1] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, Inc., New York, 1998.

[2] Makhoul, J. et al, "State of the Art in Continuous Speech Recognition," *Proceeding Natl. Acad. Sci.* USA, Vol 92 pp. 9956-9963, 1995.

[3] V. Zue, J. Glass, J. Phillips, and S. Seneff, "The SUMMIT Speech Recognition System: Phonological Modeling and Lexical Access, " *Proceedings ICASSP*, Alburquerque, 49-52, 1990.

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