14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

SUPPORT VECTOR MACHINES FOR CONTINUOUS SPEECH RECOGNITION

Jaume Padrell-Sendra[1], *Darío Martín-Iglesias*[2] *and Fernando Díaz-de-María*[2]

[1] Research Department
Applied Technologies on Language and Speech S.L (ATLAS)
jpadrell@verbio.com

[2] Signal Theory and Communications Department
Universidad Carlos III de Madrid
{dmiglesias,fdiaz}@tsc.uc3m.es

ABSTRACT

Although Support Vector Machines (SVMs) have been proved to be very powerful classifiers, they still have some problems which make difficult their application to speech recognition, and most of the tries to do it are combined HMM-SVM solutions. In this paper we show a pure SVM-based continuous speech recognizer, using the SVM to make decisions at frame-level, and a Token Passing algorithm to obtain the chain of recognized words. We consider a connected digit recognition task with both, digits themselves and number of digits, unknown. The experimental results show that, although not yet practical due to computational cost, such a system can get better recognition rates than traditional HMM-based systems (96.96% vs. 96.47%). To overcome computational problems, some techniques as the Mega-GSVCs can be used in the future.

1. INTRODUCTION

Hidden Markov Models (HMMs) are, undoubtedly, the most employed core technique for Automatic Speech Recognition (ASR). During the last decades, research in HMMs for ASR has brought about significant advances and, consequently, the HMMs are currently very accurately tuned for this application. Nevertheless, we are still far from achieving high-performance ASR systems.

Support Vector Machines (SVMs) are state-of-the-art classifiers. SVMs solution relies on maximizing the distance between the samples and the classification border. This distance is known as the margin and, by maximizing it, they are able to generalize unseen patterns. This maximum margin solution allows the SVM to outperform most nonlinear classifiers in the presence of noise, which is one of the long-standing problems in ASR. Also, SVMs don't have the convergence and stability problems typical of other classifiers as Neural Networks (NNs).

Nevertheless, the successful application of SVMs to the ASR problem requires solving three main problems. First, the variable time duration of each utterance. Second, neither the time position of each word or the number of words to be sought in the utterance are known. The last big problem is related to the size of the databases used in speech recognition, which are huge compared to the maximum number of train patterns that an SVM can deal with.

Previous works ([1] is a good example) have tried to overcome the problem of alignment using an HMM for this task, prior to classification. With this approach, however, efficiency of SVMs is limited by the errors in the segmentation stage. Other works cope with the variable time duration of the utterances embedding either, an HMM [2], or a Dynamic Time Warping algorithm [3], in the kernel of the SVM. It is not easy, however, to apply these last techniques to the problem of continuous speech because a previous segmentation in words of the utterance would be required.

Our solution to the SVM problems mentioned before consists in classifying each minimum unit of 25ms of voice (frame) as a basic class, a phone segment. With this approach we avoid the need to know where the words are and their time durations become unimportant. To go from the classification of each frame to the word chain recognition we use the same Viterbi algorithm that is used for continuous speech recognition with Hidden Markov Models (HMMs). A similar approach is taken in [4], but in that case using Neural Networks (NNs) to classify frames and to obtain their probabilities. In our case, we expect to take advantage of the robustness and higher discriminative capabilities of SVMs to get better results.

This paper is structured as follow. In the first part, sections 2 and 3, we introduce and briefly discuss our solution to the SVM problems in the context of continuous speech recognition. In the second part, section 4, we describe the experiments carried out. Finally, we conclude with a discussion in section 5.

2. CLASSIFYING EACH FRAME AS A PHONE

As we mention in the introduction we suggest applying an SVM on every frame of the speech utterance in order to determine which class the frame belongs to. For this purpose, we have as many classes as phones. In particular, for Spanish digits we can define 17 phones plus the silence. In summary, with our multiclass SVM system we will classify every individual voice frame as belonging to one of the 18 classes or phones.

2.1 Training with SVMs

There are several ways to carry out a multiclass SVM classification. In the current work we use the "one-against-one" method [5]. This method allows us to train all the system, with a maximum number of different samples for each class, with a limited computer memory. For 18 classes, this method implies to train and use $\frac{18\cdot(18-1)}{2} = 153$ SVMs, where each SVM classifies each frame between two of the possible

This work has been partially sponsored by the Regional and Spanish Government under grants GR/SAL/0472/2004 and TIC2002-02025.

phones, deciding the winning class by voting.

The number of training samples that the system is able to use becomes a practical problem for the SVM system, for both training and testing. In the training process, all the Kernels (or a high percentage of them) should be allocated in the computer memory. This fact limits the number of training samples in function of the available memory. A large training set also implies a high computational cost from the classification (test) point of view, since the number of Support Vectors (SV) increases linearly with the number of training samples.

The LIBSVM software [6] is used to train the SVMs. We use this package for two reasons. First, it implements the SMO algorithm [7] that allows a fast SVM training and with a fairly high number of samples. And second, it provides an estimated probability value for each frame and candidate phone.

2.2 Recognition with SVMs

In order to perform the recognition, we built a matrix of probabilities: one row per class (phone) and one column per frame. To obtain the chain of recognized words from this probability matrix, we use the Token Passing Model algorithm [8].

The Token Passing Model is an extension of the Viterbi algorithm typically used in continuous speech recognition to manage the uncertainty about the number of words in a sentence. Figure 1 illustrates the use of this algorithm for a very simple grammar which allows any concatenation of two Spanish words: "uno" and "tres". Classes are represented by circles, while word-ends are represented by squares. Two columns of circles are shown corresponding to two consecutive frames, i and j. The possible transitions allowed by the task grammar and explored by the Viterbi algorithm are represented either by solid or dashed lines (the mean of the line types is explained later). Each circle and transition could have an associate cost or probability. Every Viterbi node (circle) has an associated structure called *Token*. Each token stores the accumulated cost of reaching the corresponding node.

The *Token* not only stores the accumulated cost but also a *Link* to the last recognized word. The *Link* is only modified when the algorithm passes through word-ends (squares in Figure 1). The transitions among classes that modify this *Link* are represented by solid-lines, while those that do not modify it are represented by dashed-lines. Proceeding as usually in the Viterbi algorithm, only the path leading to the highest probability for every node is kept.

When the Viterbi algorithm has explored all the frames, the *Token* with a higher accumulated probability is chosen and its *Link* to the (sequence of) word-ends provides us the sequence of recognized words.

3. CLASSIFYING EACH FRAME AS A PART OF A PHONE

The definition of classes previously described can be clearly improved, since it does not take into account the time variation typically exhibited by actual phones. Some time variation can be embedded through the delta parameters (see 4.2), but better solutions should be investigated. Specifically, we have considered two alternatives: either extending the time-window covered by the parameterization (for example, considering for each time instant the concatenation of two or three consecutive features vectors), or changing the definition of classes considered in the SVM-based classification stage to deal with parts of phones.

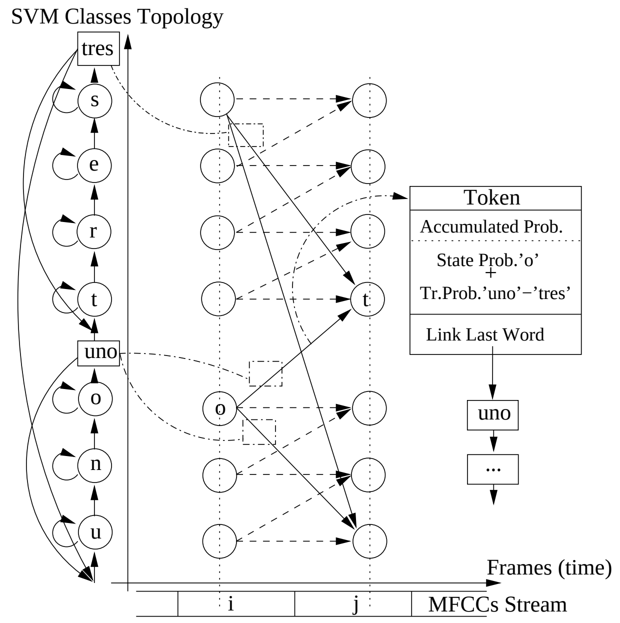

Figure 1: An illustration of the Token Passing Algorithm for a very simple grammar.

The last alternative have been chosen because it helps us to deal with another SVM-related problem: the practical limitation of the number of samples for training a single SVM. Increasing the number of classes and maintaining constant the number of samples used to train each SVM, we effectively increase the total number of samples used to train the whole system.

The natural choice consists in defining a class for the beginning of the phone, a class for the center of the phone, and finally, a class for the end of the phone. This new approach transforms our 18 initial classes into $18 \cdot 3 = 54$. In terms of the number of SVM classifiers to perform the "one-against-one" multiclass implementation, we move from 153 to 1431 SVMs.

If we use these new classes, an allowed-transition matrix should be included to actually constrain the class transitions allowed during the Viterbi-based exploration. Furthermore a "probability transition matrix" can be used instead of the previously defined "allowed-transition matrix". The transition probabilities, a_{ij}, can be estimated from the number of transitions from i to j, occurring when considering the samples in the available training set.

4. EXPERIMENTS

4.1 Database

In our experimentes we have used the subset of SpeechDat [9] containing utterances of connected digits. For this work, it has been segmented in triphones by means of an HMM-based forced alignment. From this segmentation, we obtain the locations of the phones or their three states.

SpeechDat is a telephone-speech database that includes 3496 different speakers for training (71000 files, approxi-

# Files	# Frames	% Accuracy
100	47666	82.77 [82.24,83.29]
200	98695	91.15 [90.75,91.54]
300	156530	93.18 [92.82,93.52]
400	204764	93.88 [93.54,94.21]
500	255638	94.34 [94.01,94.65]

Table 1: Baseline results achieved by the HMM-based system in terms of word recognition accuracy, for training sets consisting of 100 to 500 files. Confidence intervals are shown in brackets.

mately 100 hours of voice). For test, other 350 speakers are available (for the case of connected digits, the test set consists of 2122 files and 19855 digits). The recorded contents are varied, including phrases, digits, commands, hours, etc.

4.2 Parameterization

The voice parameterization that we have used is based on 12 Mel Frequency Cepstral Coefficients (MFCCs) plus the energy, and their first and second derivatives, known usually as the delta parameters. These MFCC are computed every 10ms over time windows of 25ms. Thus, the resulting feature vectors have 39 components. Each parameterization file is normalized in mean and variance according to the following expression:

$$\hat{x}_i[n] = \frac{x_i[n] - \mu_i}{\sigma_i + \theta}, \qquad (1)$$

where $x_i[n]$ represents the i^{th} component of the feature vector corresponding to frame n, μ_i is the estimated mean from the whole file, σ_i is the estimated variance, and θ is a constant just to avoid numerical problems (for our experiments, we have chosen $\theta = 10$).

4.3 Baseline experiment with HMMs

We use the recognition rate achieved by an HMM-based recognition system as a reference result. We use three-state phone models with 16 Gaussians per state.

Due to the practical limitation with respect to the number of training samples that the SVM software is able to manage, and for comparisons purposes, we have used in both cases (HMM- and SVM-based recognition systems) only a small part of the available training data, between 100 and 500 files of the 71000 available.

Table 1 shows the results achieved by the HMM-based system. As it can be observed, using only 100 files for training, that is the 0.14% of the available data, the HMM-based system reaches a digit recognition accuracy of 82.77%. These 100 files come only from the 4 first speakers of the 3496 available. Evidently, as the number of files for training increases, the recognition accuracy improves.

4.4 Experiments with SVMs

An SVM computes the following formula:

$$g(\overrightarrow{x}) = \sum_{i=1}^{N} \lambda_i y_i k(\overrightarrow{x}_i, \overrightarrow{x}) + b \qquad (2)$$

where $\overrightarrow{x}$ denotes the input vector, $\overrightarrow{x}_i$ are the support vectors, y_i are classification labels, λ_i are coefficients of the linear combination, b is a bias term, and $K(\overrightarrow{x}_i, \overrightarrow{x})$ is the kernel function.

The most widely used kernel function is the Gaussian Radial Basis Function (Gaussian RBF), that is the one we have chosen for our experiments:

$$k(\overrightarrow{x}_i, \overrightarrow{x}_j) = \exp\left(-\frac{||\overrightarrow{x}_i - \overrightarrow{x}_j||^2}{2\sigma^2}\right). \qquad (3)$$

Given this type of kernel, a value for the σ parameter should be selected in advance. Besides, for training, we have to choose a value for a parameter C, which establishes a compromise between error minimization and generalization capability. We have selected both parameters by Cross Validation using only a small part of the training database for several values of σ and C. In particular, we use $C = 256$ and $\sigma = 0.007812$ for all the experiments.

Table 2 shows results for three types of experiments. The first one (denoted as F1C) consists of using phones as SVM classes, the second one (F3C) uses three classes per phone and an allowed-transition matrix, and the third one (F3CMT) uses three classes per phone and a probability transition matrix.

Even when we use the smallest subset of the available files for training, i.e., 100 files, the number of frames available for training (47666 frames, see Table 1) is hard to be managed by the SVM software and conventional computational resources. In order to reduce both the training and classifying time, some maximum thresholds for the number of samples (frames) per class have been set up. Thus, a number of samples equal or less than the threshold are taken randomly from the database for each class. It is not always equal because the phonemes are not balanced in the speech, and therefore the classes will not be perfectly balanced. However, most of our experiments are approximately balanced due to the great amount of available speech samples with respect to the used thresholds.

Table 2 shows how the accuracy depends on the threshold value. To gain some insight on the threshold-based sample selection process, we analyzed which are the samples that are randomly taken out of the training set, concluding that for 100 files and a threshold of 3000 samples per class (that means using only 22532 samples out of the 47666 available) the most of the eliminated samples belonged to silences and a few of them to some vowels. Furthermore, it is interesting to notice that similar accuracies are achieved for threshold values of 3000 and 1500 (73.99% and 74.01%, respectively). However, when we identify class with phone, SVMs do not reach the HMMs results.

The results corresponding to the second experiment (F3C) are also shown in Table 2. In this experiment we have used a threshold of 1000 samples for each class, what approximately gives the same number of training samples than using a threshold of 3000 and one class per phone. With this new class structure, the recognition accuracy clearly improves: from 73.99% to 85.15%. This last result is already better than that achieved by the HMMs for the same number of training files.

The third experiment (F3CMT) also achieves to improve the accuracy with respect to the second one, from 85.15% to 87.13% (see Table 2). This result compares very favorably with that achieved by HMMs for the same number of training files: 87.13% (SVMs) vs. 82.77% (HMMs).

Finally, Table 3 shows the word recognition accuracy when we train with the whole database, 71000 files

# Files	Threshold	# Frames	# SV	% Accuracy
One Class per Phone (F1C)				
100	1500	16275	12449	74.01 [73.40,74.62]
	3000	22532	16825	73.99 [73.38,74.60]
500	1500	27247	21436	86.67 [86.19,87.14]
Three Classes per Phone (F3C)				
100	1000	20857	18005	85.15 [84.65,85.64]
Three Classes per Phone with MT (F3CMT)				
100	1000	20857	18005	87.13 [86.66,87.59]

Table 2: Three different SVM-based systems. Confidence intervals are shown in brackets.

# Frames	# SV	% Accuracy	CPU Time (hours)
HMM			
>3.5e+07	Ø	96.47 [96.21,96.72]	0.35
F3CMT			
13500	12156	93.53 [93.18,93.87]	15
27000	23606	94.89 [94.58,95.19]	36
54000	46111	95.69 [95.40,95.97]	77
108000	90007	96.22 [95.95,96.48]	141
540000	431512	96.96 [96.72,97.19]	464

Table 3: Recognition using all files available in the training database. The CPU time, in hours, consumed for every test is also provided.

(>3.5e+07 frames). For the SVM case, we always take the same number of frames per class, and we use different thresholds, increasing the total number of frames used. The difference with respect to the results shown in Table 2 is that now the selected samples for training are taken (randomly) from any file of the whole database. In all these experiments we use three classes per phone and a probability transition matrix. Looking at this results, we see that finally the SVMs reach better performance than HMMs (96.96% vs. 96.47%), and it should be noticed that the SVMs are trained using only 1.5% of the available data.

Table 3 also gives the CPU time consumed by each test. As it can be observed, the computation effort is extremely high in the case of SVMs. This is mainly due to the high number of support vectors (column labeled as #SV in Tables 2 and 3) even when we limit the number of training samples. We can also see that the number of support vectors grows linearly with the number of training samples used.

5. CONCLUSIONS

The results of this work allow us to conclude that the SVMs ca be an alternative to the HMMs in continuous speech recognition. With a very small database, 100 utterances, the SVMs improves the recognition accuracy of HMMs, and we also get a similar behavior with a large database (100 hours), although at the expense of a huge computational effort.

This last result is very interesting if we keep in mind that, due to current limitations, the SVM-based system only has used the 1.5% of the training database used by HMM-based one. We guess that a better selection of training samples (currently this selection is made randomly) for SVMs would lead to a substantial improvement.

We are trying to overcome the current limitations of our systems due to the reduced training sets that can be used. Mega-GSVCs [10] are capable of training classifiers with millions of data while keeping under control the complexity of the resulting machines. Using this kind of techniques, and as a preliminary result, we have reduced spectacularly the number of Support Vectors, moving from 46111 to 74, whilst recognition rate decreased only from 95.69% to 94.9%. With these numbers, we can expect to reduce the computation times until acceptable levels.

Another way to increase the number of frames that we can use in training is, as we have shown, increasing the number of classes considered. In this sense, we could define classes for different phonetic contexts.

Finally, in order to reduce the CPU time consumed in classification, a technique like FC-GSVC [11] or some type of Viterbi pruning could be used.

REFERENCES

[1] A. Ganapathiraju, J. Hmaker, and J. Picone, "Hybrid SVM/HMM architectures for speech recognition," in *Proc. of the International Conference on Spoken Language Processing*, 2000, vol. 4, pp. 504–507.

[2] N. Smith and M. Gales, "Speech recognition using SVMs," *Advances in Neural Information Processing Systems 14. MIT Press*, 2002.

[3] H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama, "Support vector machine with dynamic time-alignment kernel for speech recognition," in *Proc. of the Eurospeech*, 2001, pp. 1841–1844.

[4] Piero Cosi, "Hybrid HMM-NN architectures for connected digit recognition," in *Proc. of the International Joint Conference on Neural Networks*, 2000, vol. 5.

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

[6] C. C. Chang and C. J. Lin, *LIBSVM: A Library for Support Vector Machines*, 2001, Software available at http://www.csie.ntu.edu.tw/cjlin/∼libsvm.

[7] J. C. Platt, *Advances in Kernel Methods: Support Vector Learning*, chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pp. 185–208, MIT Press, 1999.

[8] S. J. Young, N. H. Russell, and J. H. S. Thornton, "Token Passing: a Conceptual Model for Connected Speech Recognition Systems," Tech. Rep., CUED Cambridge University, 1989.

[9] A. Moreno, "SpeechDat Spanish Database for Fixed Telephone Network," Tech. Rep., Technical University of Catalonia, 1997.

[10] D. Gutiérrez, E. Parrado, and A. Navia, "Mega-GSVC: Training SVMs with Millions of Data," in *Proc. of the Learning'04 International Conference*, 2004.

[11] E. Parrado, J. Arenas, I. Mora, A. Figueiras, and A. Navia, "Growing Support Vector Classifiers with Controlled Complexity," *Pattern Recognition*, vol. 36, 2003.