SVMs for Automatic Speech Recognition: A Survey

R. Solera-Ureña, J. Padrell-Sendra, D. Martín-Iglesias, A. Gallardo-Antolín, C. Peláez-Moreno and F. Díaz-de-María

Signal Theory and Communications Department
EPS-Universidad Carlos III de Madrid
Avda. de la Universidad, 30, 28911-Leganés (Madrid), SPAIN

Abstract

Hidden Markov Models (HMMs) are, undoubtedly, the most employed core technique for Automatic Speech Recognition (ASR). Nevertheless, we are still far from achieving high-performance ASR systems. Some alternative approaches, most of them based on Artificial Neural Networks (ANNs), were proposed during the late eighties and early nineties. Some of them tackled the ASR problem using predictive ANNs, while others proposed hybrid HMM/ANN systems. However, despite some achievements, nowadays, the preponderance of Markov Models is a fact.

During the last decade, however, a new tool appeared in the field of machine learning that has proved to be able to cope with hard classification problems in several fields of application: the Support Vector Machines (SVMs). The SVMs are effective discriminative classifiers with several outstanding characteristics, namely: their solution is that with maximum margin; they are capable to deal with samples of a very higher dimensionality; and their convergence to the minimum of the associated cost function is guaranteed.

These characteristics have made SVMs very popular and successful. In this chapter we discuss their strengths and weakness in the ASR context and make a review of the current state-of-the-art techniques. We organize the contributions in two parts: isolated-word recognition and continuous speech recognition. Within the first part we review several techniques to produce the fixed-dimension vectors needed for original SVMs. Afterwards we explore more sophisticated techniques based on the use of kernels capable to deal with sequences of different length. Among them is the DTAK kernel, simple and effective, which rescues an old technique of speech recognition: Dynamic Time Warping (DTW). Within the second part, we describe some recent approaches to tackle more complex tasks like connected digit recognition or continuous speech recognition using SVMs. Finally we draw some conclusions and outline several ongoing lines of research.

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. One of the most relevant problems of the HMM-based ASR technology is the loss of performance due to the mismatch between training and testing conditions, or, in other words, the design of robust ASR systems.

A lot of research efforts have been dedicated to tackle the mismatch problem; however, the most successful solution seems to be using larger databases, trying to embed in the training set all the variability of speech and speakers. At the same time, speech recognition community is aware of the HMM limitations, but the few attempts to move toward other paradigms did not work out. In particular, some alternative approaches, most of them based on Artificial Neural Networks (ANNs), were proposed during the late eighties and early nineties ([1, 2, 3, 4] are some examples). Some of them dealt with the ASR problem using predictive ANNs, while others proposed hybrid ANN/HMM approaches. Nowadays, however, the preponderance of HMMs in practical ASR systems is a fact.

In this chapter we review some of the new alternative approaches to the ASR problem; specifically, those based on Support Vector Machines (SVMs) [5, 6]. One of the fundamentals reasons to use SVMs was already highlighted by the ANN-based proposals: it is well known that HMM are generative models, i.e., the acoustic-level decisions are taken based on the likelihood that the currently evaluated pattern had been generated by each of the models that comprise the ASR system. Nevertheless, conceptually, these decisions are essentially classification problems that could be approached, perhaps more successfully, by means of discriminative models. Certainly, algorithms for enhancing the discrimination abilities of HMMs have also been devised. However, the underlying model keeps being generative.

There are other reasons to propose the use of SVMs for ASR. Some of them will be discussed later; now, we focus on their excellent capacity of generalization, since it might improve the robustness of ASR systems. SVMs rely on maximizing the distance between the samples and the classification boundary. Unlike others, such as neural networks or some modifications of the HMMs that minimize the empirical risk on the training set, SVMs minimize also the structural risk [7], which results in a better generalization ability. In other words, given a learning problem and a finite training database, SVMs properly weight the learning potential of the database and the capacity of the machine.

The maximized distance, known as the margin, is the responsible of the outstanding generalization properties of the SVMs: the maximum margin solution allows the SVMs to outperform most nonlinear classifiers in the presence of noise, which is one of the longstanding problems in ASR. In a noise-free system, this margin is related to the maximum distance a correctly classified sample

should travel to be considered as belonging to the wrong class. In other words, it indicates the noise that added to the clean samples is allowed into the system.

Nevertheless, the use of SVMs for ASR is not straightforward. In our opinion, three are the main difficulties to overcome, namely: 1) SVMs are originally static classifiers and have to be adapted to deal with the variability of duration of speech utterances; 2) the SVMs were originally formulated as a binary classifier while the ASR problem is multiclass; and 3) current SVM training algorithms are not able to manage the huge databases typically used in ASR; in spite of the appearance of techniques as Sparse SVM, the number of training samples is still limited to a few thousands.

In this Chapter we will review the solutions that during the last years have been proposed to solve the mentioned problems. Nowadays, it can be said that SVMs have been successfully used in simple ASR tasks, especially in presence of noise. On the other hand, the research work focused on more complex task is still incipient, though the results are encouraging.

This Chapter is organized as follows. Section 2 briefly reviews the ANN- and hybrid ANN/HMM-based approaches proposed during the late eighties and early nineties. First, some of the difficulties of using ANNs for ASR (that SVMs share) are revealed. Later, as a consequence of the study of the hybrid systems, some of HMM limitations are illustrated and how ANNs can be used to complement HMMs is discussed (again the lessons apply to SVMs). Section 3 summarizes the SVM fundamentals, emphasizing those aspects relevant from the ASR perspective. Section 4 is the core of the Chapter. The expected advantages of SVMs in ASR are reviewed. The limitations to be overcome are discussed. The most relevant research works dealing with SVMs for ASR are briefly described. For that purpose, the different contributions are organized in two subsections depending on the ASR task complexity: first, isolated-phone, -letter or -word recognition and after connected-words or continuous speech recognition. Finally, some conclusions are drawn and future lines of research are outlined in Section 5.

2 ANNs for ASR

In next paragraphs, we briefly introduce the application of Artificial Neural Networks (ANNs) to the speech recognition problem. This section does not try to be an exhaustive review of this matter. On the contrary, its aim is to outline the main alternatives proposed for the integration of ANNs into ASR systems in order to illustrate their similarities with the use of SVMs for the same purpose, especially in the context of hybrid HMM-based ASR systems.

During the last two decades some alternative approaches to HMMs, most of them based on ANNs, have been proposed for ASR as an attempt to overcome the limitations of the HMMs. ANNs represent an important class of discriminative techniques, very well suited for classification problems. In particular, ANNs exhibit several properties that have motivated their application to the implicit pattern classification problem in ASR, namely [4]:

- They learn according to discriminative criteria. Although other classifiers like HMMs can be trained in a discriminative framework, ANN training is inherently discriminative.
- ANNs are the universal approximators, i.e., they can approximate any continuous function with a simple structure.
- ANNs do not require strong assumptions about the underlying statistical properties of the input data and the functional form of the output density. On the contrary, HMMs usually assume that successive acoustic vectors are uncorrelated and follow a Gaussian (or mixture of Gaussians) distribution.

Despite of the good performance of ANNs on static classification problems, they present notable limitations to deal with the classification of time sequences as is the case of speech signals. In fact, this has been one of the fundamental problems to solve in the application of ANNs to speech recognition tasks.

2.1 ANN-based ASR systems

In order to deal with the time sequence classification problem, the first ANN-based ASR systems pursued the adaptation of the neural network architecture to the temporal structure of speech. In this context, two different classes of neural networks which consider the correlation between the temporal structures in the speech patterns were proposed: Time-Delay Neural Networks (TDNNs) [8] and Recurrent Neural Networks (RNNs) [9].

TDNNs can be considered as a special type of the well-known Multilayer Perceptron (MLP) in which input nodes integrate shift registers (or time delays). This way, the TDNN training is performed over a time sequence of acoustic vectors and the network is capable of incorporating a local acoustic context into the whole process. RNNs are a generalization of the MLP network in which feedback connections are allowed. As a consequence, the network behavior is based on its history providing a mechanism to model time sequence patterns.

Although these systems have shown to achieve good results on phoneme or isolated word recognition tasks, ANNs have not been successful on more complex tasks as continuous speech recognition. The main reason for this lack of success has been their inability to model the time variability of the speech signal even when recurrent structures are used.

2.2 Hybrid ANN/HMM-based ASR systems

To overcome these difficulties, several researchers have proposed the so-called Hybrid ANN/HMM-based ASR systems. The basic idea underlying these schemes is to combine HMMs and ANNs into a single system to get profit from the best properties of both approaches: the ability of HMMs to model the time variability of the speech signal and the discrimination ability provided by ANNs. Following this principle, different classes of hybrid ANN/HMM systems

have been developed. In next paragraphs, we briefly describe some of the most relevant ones. A complete survey about this subject can be found in [10].

The most common approach to hybrid systems is the initially proposed in [11, 4] in which an ANN is used to estimate jointly all the HMM state emission probabilities. Several types of neural networks have been used for this purpose: MLPs [4], RNNs [12] and even Radial Basis Function (RBF) networks [13].

Other approaches for speech recognition use Predictive Neural Networks, one per class, to predict a certain acoustic vector given a time window of observations centered in the current one [2], [3]. This way Predictive Neural Networks capture the temporal correlations between acoustic vectors.

Finally, in the hybrid ANN/HMM system proposed in [14], ANNs are trained to estimate phone posterior probabilities and these probabilities are used as feature vectors for a conventional GMM-HMM recognizer. This approach is called Tandem Acoustic Modeling and it achieves good results in context-independent systems.

Numerous studies show that hybrid systems achieve comparable recognition results than equivalent (with a similar number of parameters) HMM-based systems or even better in some tasks and conditions. Also, they present a better behavior when a little amount of training data is available. However, hybrid ANN/HMM have not been yet widely applied to speech recognition, very likely because some problems still remain open, for example: the design of optimal network architectures or the difficulty of designing a joint training scheme for both, ANNs and HMMs.

3 SVM fundamentals

3.1 SVM formulation

A SVM is essentially a binary nonlinear classifier capable of guessing whether an input vector $\mathbf{x}$ belongs to a class 1 (the desired output would be then $y = +1$) or to a class 2 ($y = -1$). This algorithm was first proposed in [15] in 1992, and it is a nonlinear version of a much older linear algorithm, the optimal hyperplane decision rule (also known as the generalized portrait algorithm), which was introduced in the sixties.

Given a set of separable data, the goal is to find the optimal decision function. It can be easily seen that there is an infinite number of optimal solutions for this problem, in the sense that they can separate the training samples with zero errors. However, since we look for a decision function able to generalize for unseen samples, we can think on an additional criterion to find the best solution among those with zero errors. If we knew the probability densities of the classes, we could apply the maximum a posteriori (MAP) criterion to find the optimal solution. Unfortunately, in most practical cases this information is not available, so we can adopt another simpler criteria: among those functions without training errors, we will choose that with the *maximum margin*, being this margin the distance between the closest sample and the decision boundary

defined by that function. Of course, optimality in the sense of maximum margin does not imply necessarily optimality in the sense of minimizing the number of errors in test, but it is a simple criterion that yields to solutions which, in practice, turn out to be the best ones for many problems [16].

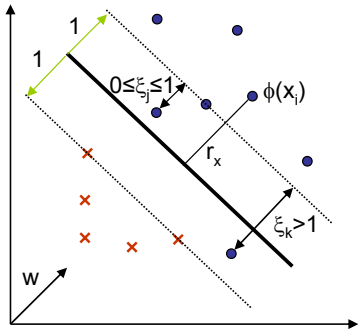

Figure 1: Soft-margin decision

As can be inferred from the Figure 1, the nonlinear discriminant function $f(\mathbf{x}_i)$ can be written as:

$$f(\mathbf{x}_i) = \mathbf{w}^T \cdot \phi(\mathbf{x}_i) + b, \tag{1}$$

where $\phi(\mathbf{x}_i) : \Re^n \mapsto \Re^{n'}$, $(n << n')$, is a nonlinear function which maps the vector $\mathbf{x}_i$ into what is called a *feature space* of higher dimensionality (possibly infinite) where classes are assumed to be linearly separable. The vector $\mathbf{w}$ represents the separating hyperplane in such a space. It is worth noting that the meaning of *feature space* here has nothing to do with the space of the speech features that within the kernel methods nomenclature belong to the *input space*.

On the other hand, r_x is the distance between the transformed sample $\phi(\mathbf{x}_i)$ and the separating hyperplane, and $\| \mathbf{w} \|$ the Euclidean norm of $\mathbf{w}$. We call *support vectors* those closest to the decision boundary. These vectors define the margin and are the only samples that are needed to find the solution. Thus, we have that for every sample $\mathbf{x}_i$, $r_x = f(\mathbf{x}_i)/ \| \mathbf{w} \|$. Hence, the goal to find the optimum classifier is achieved by minimizing $\| \mathbf{w} \|$ with the restriction of all samples being correctly classified, i.e.:

$$y_i \left(\mathbf{w}^T \cdot \phi(\mathbf{x}_i) + b \right) \geq 1. \tag{2}$$

This can be formulated as a problem of quadratic optimization:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2} \| \mathbf{w} \|^2, \\ \text{subject to} \quad & y_i(\mathbf{w}^T \cdot \phi(\mathbf{x}_i) + b) \geq 1 \end{aligned}$$

In order to get a classifier with a better generalization ability and capable of handling the non-separable case, we should allow a number of misclassified data. This is accomplished by introducing a penalty term in the function to be minimized:

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i} \quad & L_P = \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{N} \xi_i, \\ \text{subject to} \quad & y_i(\mathbf{w}^T \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \text{ for } i = 1, \cdots, N, \end{aligned} \tag{3}$$

where $\mathbf{x}_i \in \Re^n$ $(i = 1, \ldots, N)$ are the training vectors corresponding to the labels $y_i \in \{\pm 1\}$, and the variables ξ_i are called *slack variables* and allow a certain amount of errors that contribute to obtain solutions in the non-separable case. ξ_i verifies $0 \leq \xi_i \leq 1$ for those samples well classified but inside the margin, and $\xi_i > 1$ for those samples wrongly classified. The C term, on the other hand, expresses the trade-off between the number of training errors and the generalization capability.

This problem is usually solved introducing the restrictions in the function to be optimized using Lagrange multipliers, leading to the maximization of the Wolfe dual:

$$\begin{aligned} \max_{\alpha_i} \quad & L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \phi^T(\mathbf{x}_i) \phi(\mathbf{x}_j), \\ & \text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \ \text{ and } \ 0 \leq \alpha_i \leq C. \end{aligned} \tag{4}$$

This problem is quadratic and convex, so its convergence to a global minimum is guaranteed using quadratic programming (QP) schemes. The resulting decision boundary $\mathbf{w}$ will be given by:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i). \tag{5}$$

According to (5), only vectors with an associated $\alpha_i \neq 0$ will contribute to determine the weight vector $\mathbf{w}$ and, therefore, the separating boundary. These are the support vectors that, as we have mentioned before, define the separation border and the margin.

Generally, the function $\phi(\mathbf{x})$ is not explicitly known (in fact, in most of the cases its evaluation would be impossible as long as the feature space dimensionality can be infinite). However, we do not actually need to know it, since we only need to evaluate the dot products $\phi^T(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ which, by using what has been called the *kernel trick*, can be evaluated using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$.

Many of the SVM implementations compute this function for every pair of input samples producing a *kernel matrix* that is stored in memory.

By using this method and replacing $\mathbf{w}$ in (1) by the expression in (5), the form that a SVM finally adopts is the following:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b. \tag{6}$$

The most widely used kernel functions are:

- the simple *linear* kernel

$$K_L(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j; \tag{7}$$

- the *radial basis function kernel* (RBF kernel),

$$K_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right), \tag{8}$$

  where γ is proportional to the inverse of the variance of the Gaussian function and whose associated feature space is of infinite dimensionality; and

- the polynomial kernel

$$K_P(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \mathbf{x}_i^T \cdot \mathbf{x}_j\right)^p, \tag{9}$$

  whose associated feature space are polynomials up to grade p, and

- the sigmoid kernel

$$K_{SIG}(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(a\mathbf{x}_i^T \cdot \mathbf{x}_j + b\right), \tag{10}$$

It is worth mentioning that there are some conditions that a function should accomplish to be used as a kernel. These are often denominated KKT (Karush-Kuhn-Tucker) conditions [17] and can be reduced to check the kernel matrix is symmetrical and positive semi-definite.

3.2 Pros and cons of SVMs

The reason that makes SVMs more effective in many applications than other methods based on linear discriminants is its learning criterion. The goal of any classifier must be minimizing the number of misclassifications in any possible set of samples. This is known as Risk Minimization (RM). However, in typical classification problems we only have a limited number of samples available (in some cases we can have an unlimited number of samples but, anyway, we only can deal with a subset), and so, all we can do is trying to minimize the number

of misclassifications within the training set. This is known as Empirical Risk Minimization (ERM), and most classifiers base their learning process on it.

However, having the classifier with the best ERM is not enough (or even desirable). The complexity of the classifiers normally must be fixed a priori, and so, we can choose a too simple structure unable to model correctly the classification boundaries of our problem, or a too complex one, overfitted to our training set and unable to generalize to unseen samples. This is known as Structural Risk, and a good classifier must maintain a compromise between the ERM and the SRM (Structural Risk Minimization).

In SVMs, we do not need to previously fix the complexity of the resultant machine, but there is a parameter (the C in equation 3) which establishes this compromise between ERM and SRM. Unfortunately, there is no method to know a priori the most adequate value for this parameter, so we must find it by means of a search process.

Other advantages of SVMs are:

- They have a unique solution and its convergence is guaranteed (the solution is found by minimizing a convex function). This is an advantage compared to other classifiers as ANNs that often fall in local minima or does not converge to a stable version.

- The solution is that with maximum margin, what makes these machines robust and, in our opinion, very well suited for applications as ASR in noisy environments.

- Since in the minimization process only the kernel matrix is involved, they can deal with input vectors of very high dimensionality, as long as we are capable of calculating their corresponding kernels. In practice, they can deal with vectors of thousands of dimensions.

Among the disadvantages, we can highlight the following:

- Most implementations of SVM algorithm require to compute and store in memory the complete kernel matrix of all the input samples. This task have a space complexity $O(n^2)$, and is one of the main problems of these algorithms that prevent their application on very large speech databases. Most implementations allow us to work with some thousands of samples. However, some modifications of the algorithm are being developed which would allow us to work with millions of samples [18].

- The optimality of the solution found can depend on the kernel we have used, and there is not a method to know a priori which will be the best kernel for a concrete task. Although kernels as RBF are considered *universal*, it is still necessary to perform a grid-based search to fix all the parameters of the SVM.

- As we have mentioned, the best value for the parameter C is also unknown a priori.

- Like ANNs, the input vectors of an SVM with the formulation we have seen, must have a fixed size. This is a problem in speech recognition where each sequence to be recognized has a different duration. There are some solutions to this problem that we will discuss later.

However, despite these troubles, SVMs are attractive enough to be used in a variety of applications and, specifically, in speech recognition.

4 SVMs for ASR

As already discussed in the Introduction and in the previous section, SVMs are state-of-the-art tools for solving classification problems that seems to be very promising from the speech recognition perspective. They offer a discriminative solution to the pattern classification problem involved in ASR. Furthermore, the maximum margin SVM solution exhibits an excellent generalization capability, what might notably improve the robustness of ASR systems.

In fact, the improved discrimination ability of SVMs has attracted the attention of many speech technologists. Though this paper focuses on speech recognition, it is worth noticing that SVMs have already been employed in speaker identification [19] and verification [20], or to improve confidence measurements that can help in dialogue systems [21], among other applications.

However, its application to ASR is by no means straightforward. Here follows a review of the most important problems that has motivated the structure of the present section.

- *The variable time duration of the speech utterances*: The Automatic Speech Recognition involves the solution of a pattern classification problem. However, the variable time duration of the speech signals has prevented the ASR from being approached as a simple static classification problem. In fact, this has been for many decades one of the fundamental problems faced by the speech processing community and the main responsible for the success of the HMMs. The main problem stems from the fact that conventional kernels can only deal with (sequences of) vectors of fixed length. Standard parameterization techniques, on the other hand, generate variable length sequences of feature vectors depending on the time duration of each speech utterance.

  Different approaches have been proposed to deal with the variable time duration of the acoustic speech units. Basically, solutions can be divided into three groups: 1) the ones that aim at performing a previous dimensional (time) normalization to fit the SVM input; 2) those that explore string-related or normalizing kernels [5] to adapt the SVMs to make them able to use variable dimension vectors as inputs; and 3) those that avoid this problem by working in a framewise manner. As we will see later in section 4.2, the latter is specially well suited for continuous speech recognition while the first two are more appropriate for lower complexity tasks and will be addressed in section 4.1.

- *Multiclass SVMs*: ASR is a multiclass problem, whereas in the original formulation an SVM is a binary classifier. Although some of the proposed approaches to multiclass SVMs make a reformulation of the SVM equations to consider all classes at once, this option is very computationally expensive. A more usual approach to cope with this limitation involves combining a number of binary SVMs to achieve the multiclass classifier by means of a subsequent voting scheme. Two different versions of this method are usually considered. The first consists of comparing each class against all the rest (*1-vs-all*), while in the second each class is confronted against all the other classes separately (*1-vs-1*). Although the number of SVMs is greater for the *1-vs-1* approximation (namely, $\frac{k(k-1)}{2}$ vs. k SVMs, with k denoting the number of classes), the size of the training set needed for each SVM in the *1-vs-1* solution leads to a smaller computational effort with comparable accuracy rates [22].

- *The size of the databases*: most SVM implementations do not allow to deal with the huge databases typically used in medium- and high-complexity ASR task.

Having reviewed the fundamental challenges we now devote the next subsections to the exposition of the main solutions described in the literature, from the most simple tasks, such as isolated phonemes, letters or words recognition (low-complexity ASR tasks) to approaches to connected digits and continuous speech recognition (medium-complexity ASR tasks).

4.1 Isolated-word recognition

In this subsection we summarize some of the most relevant approaches to isolated unit (phonemes, letters or words) recognition by means of SVMs. We will distinguish between solutions that involve a preprocessing of the speech feature sequences and SVM-specific solutions capable of working with samples of variable dimensionality. The later are most of the times based on what is called *sequence kernels* that, in our opinion, show a great potential even for the their application to more complex task. Therefore, we will provide a more detailed overview of two instances of those kernels, namely, the DTAK and Fisher kernels.

4.1.1 Preprocessing of the speech feature sequences

When dealing with this type of ASR tasks, the main problem of SVM-based approaches is the time normalization of the different utterances of the acoustic units (to get a fixed-dimension input space). On the other hand, the complexity of the SVM implementation (training or testing) is not a problem because the lexicon is usually quite limited.

Several authors use different variations of the the so-called *triphone model approach*. This model is motivated by the three-state HMMs used in most state-of-the-art speech recognition systems that amounts to assume that the speech

segments (phones or triphones in most cases) can be decomposed into a fixed number of sections. The first and third sections model the transition into and out of the segment, whereas the second section models the stable portion. The main variants of this approach are summarized below:

- In [23] they show significant improvement in performance on a static pattern classification task based on the Deterding vowel data as well as on a continuous alphadigit one (OGI Alphadigits). The vector resulting from the concatenation of the three segments corresponding to the triphone model is augmented with the logarithm of the duration of the phone instance to explicitly model the variability in duration. The composite feature vectors are based on the alignments from a baseline three-state Gaussian-mixture HMM system. SVM classifiers are trained on these composite vectors, and recognition is also performed using these segment-level composite vectors. They have also used this model in a large vocabulary conversational speech task (Switchboard) as we will review in next subsection.

- In [24] they use SVMs for two different tasks, namely: Thai tone and Thai vowel recognition, using different feature length normalization procedures for each of them. The first one is Thai tone recognition in which they try to classify the five different lexical tones in that language: mid, low, falling, high and rising. A fixed number of measures of the pitch evolution is chosen in this case. However for the classification of Thai vowels they also divide each vowel into three regions.

- In [25], the authors evaluate the performance of SVMs showing advantages when compared with GMM (Gaussian Mixture Models) in both vowel-only and phone classification tasks. It is worth noting that a significant difference is observed in the problem of length adaptation between these two tasks. In the vowel case, it is acknowledged that regardless of the duration of each utterance, the acoustic representations are almost constant. Therefore simple features as the formant frequencies or LPC coefficients corresponding to any time window are representative of the whole sequence. However, the representation of the variations taking place in non-vowel utterances is essential for obtaining an adequate input to SVMs. Thus, again the triphone model approach has been applied in this case, segmenting the number of frames obtained for each phone into three regions in the ratio 3-4-3 and subsequently averaging the features corresponding to the resulting regions.

- Similar distinctions have been observed in [26], where a comparison between the performance of classical HMMs and SVMs as sub-word units recognition is assessed for two different languages: 41 monophone units are classified in a Japanese corpus and 86 consonant-vowel units are considered for an Indian language. In this case, two different strategies have been devised to provide the SVMs with a fixed-length input: for the Japanese

monophones, a similar technique to that proposed in [25] has been used. The frames comprising each monophone have been divided into a fixed number of segments. An averaged feature vector is then obtained for each segment. Each feature vector is subsequently concatenated to those resulting from other segments to form input vector for the SVM classifier. For the Indian consonant-vowel classification, however, a different approach has been designed to account for the variations of the acoustic characteristics of the signal during the consonant-vowel transition. In this case the fixed length patterns are obtained by linearly elongating or compressing the feature sequence duration. For both Indian and the previously mentioned Japanese tasks the SVMs have shown a better performance than HMMs with the standard MFCCs (Mel-Frequency Cepstral Coefficients) [27] plus energy and delta and acceleration coefficients.

In [28] several ways of preprocessing the speech sequence to obtain a fixed dimension vector are analyzed for a noisy digit recognition task. Two methods of sequence uniform resampling are assessed performing variations on the size of the analysis window and the frame period: a variable window size method that makes it possible to include the whole digit utterance for a given number of windows per digit by adjusting the size of the window to the digit duration, and a fixed window size one, that maintains the window size around a fixed number of analysis instants regardless of the coverage of the digit it does.

In [29] their primary goal is to solve the problem of the computational complexity of the SVM classical formulation by using an alternative Lagrangian one on the TIMIT database. Their feature representation uses the previously explained variable window size method using different window lengths based on the duration of the phoneme being classified. Therefore they concatenate 5 windows of the same size chosen from the set {32, 64, 128, 256, 400} covering the whole phoneme.

Another possible solution is showed in [28, 30, 31], where the non-uniform distribution of analysis instants provided by the internal states transitions of an HMM with a fixed number of states and a Viterbi decoder is used for dimensional normalization. The rationale behind this proposal is that the uniform resampling methods are produced without any consideration about the information (or lack of information) that speech analysis segments were providing. Selecting the utterance segments in which the signal is changing, it is hoped that a bigger amount of information is preserved in the feature vector.

Related to the previous approach, in [32] they acknowledge the fact that the classification error patterns from SVM and HMM classifiers can be different and thus their combination could result in a gain in performance. They assess this statement on a classification task of consonant-vowel units of speech in several Indian languages obtaining a marginal gain by using a sum rule combination scheme of the two classifiers evidences. As for feature length normalization they select segments of fixed duration around the vowel onset point, i.e., the instant at which the consonant ends and the vowel begins.

4.1.2 Isolated-digit recognition with DTAK-SVMs

This method was introduced in [33] and [34], and belongs to the family of methods based on *sequence kernels*, which try to solve the problem of different length sequences by adapting the kernel of the SVM to one capable of working with samples of variable dimensionality. This seems to be a more natural approach than performing a previous segmentation.

Summarizing, this technique uses as a kernel the score obtained by means of a Dynamic Time Warping (DTW) algorithm. DTW algorithms were one of the first techniques used in speech recognition and they were widely used in the 70s [35].

DTW measures the distance between a target signal and a template, expanding or contracting the temporal axis of the target to find the *path* or *warping function* which maximizes the similarity between the two signals (Figure 2). The distance of the signals is computed at each instant along the warping function, and the final score given by the algorithm is the accumulated similarity. Any metric can be used to compute this distance but usually the Euclidean is employed. In the case of DTAK, the inner product is used and therefore this distance can be interpreted as a linear kernel that is employed internally for the computation of the DTAK Kernel. With such an interpretation, it is now possible to substitute this distance metrics for the one provided by non-linear kernels such as RBF as we will introduce further on.

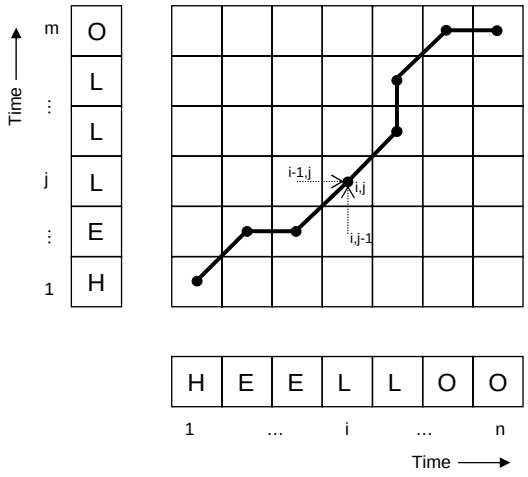

Figure 2: Dynamic Time Warping

Specifically, for the computation of the linear kernel we use the following procedure: if X and Y are the two sequences of feature vectors to be compared, and $\psi_I(k)$ and $\psi_J(k)$ are warping functions which normalize the temporal axis of the sequences in the instant k, we must find the solution to the new *inner product*:

$$
\begin{aligned}
K_{DTA}(X,Y) = X \circ Y \;\; &= \;\; \max_{\psi_I,\psi_J} \frac{1}{M_\psi} \sum_{k=1}^{L} m(k) \mathbf{x}_{\psi_I(k)}^T \cdot \mathbf{y}_{\psi_J(k)}, \\
&\text{subject to } \; 1 \leq \psi_I(k) \leq \psi_I(k+1) \leq |X|, \\
&\qquad\quad 1 \leq \psi_J(k) \leq \psi_J(k+1) \leq |Y|,
\end{aligned} \tag{11}
$$

where $|\cdot|$ denotes the length of the sequences, M_ψ is a normalization factor which normally has the value $M_\psi = |X|+|Y|$, L is a normalized length that can be either $|X|$, $|Y|$ or arbitrary positive integer, and $m(k)$ is a non negative scale factor which gives more importance to some particular "steps" in the "path".

This optimization problem is normally solved by means of *dynamic programming*, using the following recursive equation:

$$
D(i,j) = \max \left\{ \begin{array}{c} D(i-1,j) + \mathbf{x}_i^T \cdot \mathbf{y}_j, \\ D(i-1,j-1) + 2\mathbf{x}_i^T \cdot \mathbf{y}_j, \\ D(i,j-1) + \mathbf{x}_i^T \cdot \mathbf{y}_j. \end{array} \right. \tag{12}
$$

where the scale factor '2' favors translations along the diagonal, which should be the most probable ones. Therefore, the DTA Kernel gets reduced to,

$$
K_{DTA}(X,Y) = X \circ Y = D(|X|,|Y|)/(|X|+|Y|) \tag{13}
$$

It is worth mentioning that in contrast with the classical template-based ASR solutions where the difficulty of finding an appropriate template was the main drawback that lead to the supremacy of the model based approaches like HMM, the DTAK solution automatically finds the best reference templates using the max-margin criterion.

Effectively, if we look at equation (5) in section 3, we see that only those templates with an associated $\alpha_i \neq 0$ will be relevant and will contribute to determine the separating boundary. Only a few templates will have a non-zero α_i, and these will be the closest to the decision function. Now, the *support vectors* are *support sequences* or templates.

Furthermore, the algorithm not only selects those appropriate templates that define the decision boundary but the number of them that minimise the structural risk, and this is accomplish by giving an appropriate value to the parameter C in equation (3). Unfortunately, we do not have a method to calculate the best value for this parameter *a priori*, so we must resort to cross-validation.

With the previous formulation it is now easy to consider the generalization that allows us to find the separating border in a higher dimension space (the feature space) by means of a non-linear kernel like an RBF. We have said that, basically, DTAK consists in using DTW as the kernel of an SVM. However, a generalization consisting in performing the time-warping in the feature space can be considered. In other words, in equation (11), we could use a kernel function (for example, an RBF) instead of a conventional dot product and the DTAK kernel would have the following form:

$K_{sDTA}(X,Y)$

$$= \phi(X) \circ \phi(Y) = \max_{\psi_I, \psi_J} \frac{1}{M_\psi} \sum_{k=1}^{L} m(k) K_{RBF}(\mathbf{x}_{\psi_I(k)}, \mathbf{y}_{\psi_J(k)}). \quad (14)$$

Now, we have to demonstrate that K_{sDTA} fulfills the KKT conditions. As we mentioned in section 3, the only thing we have to prove is that K_{sDTA} is symmetrical and positive semidefinite. The former is obvious, since the warping function is the same if we interchange the sequences X and Y. Regarding the latter, we must demonstrate that:

$$\mathbf{u}^t \mathbf{K}_s \mathbf{u} \geq 0 \ \ \forall \mathbf{u}. \quad (15)$$

This is easily proved if we consider that DTW is the (weighted) sum of the inner products (kernels) of the vectors composing the sequences X and Y at the instants defined by the optimal warping function $\psi^*(k)$. That is (omitting the scale factors):

$$\mathbf{K}_s = \mathbf{K}_{(1)} + \cdots + \mathbf{K}_{(L)}, \quad (16)$$

where $\mathbf{K}_{(k)}$ is the kernel at the instant defined by $\psi^*(k)$. So,

$$\begin{aligned} \mathbf{u}^t \mathbf{K}_s \mathbf{u} &= \mathbf{u}^t (\mathbf{K}_{(1)} + \cdots + \mathbf{K}_{(L)}) \mathbf{u} \\ &= \mathbf{u}^t \mathbf{K}_{(1)} \mathbf{u} + \cdots + \mathbf{u}^t \mathbf{K}_{(L)} \mathbf{u} \\ &\geq 0, \end{aligned} \quad (17)$$

since $\mathbf{K}$ is a valid kernel and, therefore, positive semidefinite.

Experimental Results

Results for the well-known SpeechDat-4000 database are presented in [31]. The whole database is not used: specifically, only the isolated-digit utterances are used for the experiments. The reported results depend on the noise level: on the one hand, the DTAK-based system achieves excellent performance, clearly superior to that achieved by the HMM-based system, for low SNRs; on the other hand, it incurs in some performance losses for high SNRs. The improvements due to DTAK (using either linear or RBF kernels) with respect to HMMs are statistically significant for white noise at 3, 6 and 9 dB and for F16-plane noise at 3 and 6 dB. On the contrary, the HMM-based system outperforms the DTAK-based one in clean conditions and several noisy cases at 12 dB. In the remaining conditions, which correspond to medium SNRs, the system performances do not exhibit statistically significant differences. Please refer to [31] for more details about the DTAK-based system and the experimental setup.

In summary, the reported results [31] show that SVMs exhibit a robust behaviour, as expected. In particular, the DTAK-based system turns out to be

effective in noisy scenarios. In fact, the advantage due to the DTAK algorithm is higher as the noise conditions worsen. On the other hand, direct application of DTAK-based systems to continuous speech recognition is by no means straightforward, as the length of the sequences in eq. (13) must be known. In our opinion, some alternative segmentation techniques such as that proposed in [36], should be revisited to deal with this limitation.

4.1.3 Other types of sequence kernels: the Fisher kernels

DTAK is an instance of the so called *sequence kernels* that try to solve the problem of the different duration of the input sequences by looking for kernels capable of working with vectors of variable dimensionality. In this section we outline one of the most popular ones: the *Fisher kernel* and all its derivative family.

The Fisher kernel was first used in the biology field, in the context of DNA and protein sequence analysis [37], although there are also some interesting results in the field of speech recognition. Thus, in [38, 39, 40], this method is evaluated on a speaker-independent isolated letter task, outperforming the standard HMMs. Much more promising, however, are the results in speaker verification. In [41], the presented SVM system outperforms up to 34% the rates obtained with a GMM model.

The idea behind this method is to use as a kernel a score function computed by using the a posteriori probabilities of the observations obtained with a generative model (GMM, HMM...). Therefore the Fisher kernels takes advantage of the capability of the generative models to work with sequences of different lengths.

Let $P(X \mid \theta)$ be the a posteriori probability obtained with a generative model with parameters θ. The set of all the $P(X \mid \theta)$ corresponding to all the different $\theta \in \Theta$ (being Θ the set of all possible parameters of the model), forms a Riemann manifold M_Θ. In such a space, the inner product is given by $U_{X_i}^T F^{-1} U_{X_j}$, where $F = E_X\left[U_X U_X^T\right]$ is the Fisher information matrix, and $U_X = \nabla_\theta \log P(X \mid \theta)$ is named the Fisher score .

Summarizing, the steps to calculate the Fisher kernel are:

- Get $P(X \mid \theta)$ from a generative model.
- Calculate $U_X = \nabla_\theta \log P(X \mid \theta)$. This can be quite complex but the steps to obtain this expression from an HMM are especified in [39].
- Calculate $F = E_X\left[U_X U_X^T\right]$ (in some texts, this matrix is approximated by the identity, or by $\sigma^2 I$ therefore implying a conventional inner product and a Euclidean space).
- $K(X_i, X_j) = U_{X_i}^T F^{-1} U_{X_j}$.

It is easy to demonstrate that K is symmetrical and positive semi-definite and, hence, a kernel, since F fulfils those conditions.

In the same way that in the conventional kernels it is also possible to modify this kernel to obtain RBF or polynomial kernels. For example, the polynomial Fisher kernel would be:

$$\tilde{K} = (1 + K(X_i, X_j))^p \tag{18}$$

In [37] it is demonstrated that a discriminative classifier based on the Fisher kernel is at least so good as the Maximum A Posteriori (MAP) classifier of the generative model associated.

We can further generalize the Fisher kernel by substituting the logarithm and ∇ operators of the score for other types of operations. For example a modification specially useful in speaker verification, employs the logarithm of the ratio between the a posteriori probabilities generated by two different models.

A final remark concerning both types of sequence kernels we have presented is that the support vectors they compute act as templates against which the incoming sequences are compared. For DTAK kernels these support vectors were particular sequences and here they are scores. This templates, however, are not the most representative instance of a class, as in the conventional template based pattern recognition but are the smaller set of vectors that we can combine to define the border between two classes.

However, the main problem that, thought they are capable of comparing different duration acoustic units, the boundaries of these units must be previously determined. This is their major drawback that prevents their application to continuous speech recognition.

4.2 Connected-digit and continuous speech recognition

Either connected-word recognition or continuous speech recognition are obviously more complex tasks than isolated-word recognition. In particular, the successful application of SVMs to more complex ASR tasks requires solving two additional problems. First, neither the time position of each word nor the number of words to be sought in the utterance are known. And second, the more complex it is the ASR task, the larger is the speech data base required for the design of the system; consequently, the size of the databases used in more complex tasks turns out to be huge compared to the maximum number of training samples that a SVM can deal with. Nevertheless, the very valuable characteristics of SVM classifiers have encouraged several authors to try to solve these problems.

As briefly mentioned in a previous section, some authors [42] have tried to overcome the problem of the variability of duration of speech utterances using HMMs to perform a time segmentation prior to classification. Other works cope with the variability of duration of speech utterances by embedding either an HMM [38] or a Dynamic Time Warping algorithm [33] in the kernel of the SVM. It is not easy, however, to apply these last two techniques to the problem of continuous speech because a previous word (or phoneme) segmentation of the utterance is still required. Another solution to overcome the mentioned

difficulties is proposed in [43]. This method consists in classifying each frame of voice as belonging to a basic class (a phone) and using the Token Passing algorithm [44] to go from the classification of each frame to the word chain recognition. This is a similar approach to that presented in [45] by Cosi. The main difference is that Cosi uses Neural Networks (NNs) instead of SVMs.

In this section the approaches due to Ganapathiraju [42], who proposed a hybrid HMM/SVM system, and Padrell [43], who presented a pure SVM-based ASR system, are explained in detail.

Although it will not be described in this Chapter, it is worth to briefly mention a segmentation method for continuous speech presented in [46]. In particular, articulatory features are used to segment speech into broad manner classes using the probability-like outputs of SVMs to perform the classification every 5 ms over a 10 ms duration frame. They found that for this task, SVMs perform significantly better than HMM.

4.2.1 Hybrid HMM/SVM-based continuous speech recognition [42]

In this case the HMMs are used to generate phonetic level alignments that are treated individually by the SVM to perform phoneme identification. Since each segment will have a different duration, some method is needed to convert them to fixed length vectors. These methods were revised in 4.1. Here we illustrate with some more detail the method proposed in [42] for a continuous speech recognition task. These authors suggest dividing the segment into three regions according to a pre-established proportion; thus, the vectors of the parameterized signal can be split into three groups according to a distribution of 30%-40%-30%. Then the vectors into every region are averaged and finally concatenated as depicted in Figure 3.

4.2.2 SVM-based continuous speech recognition [43]

The hybrid HMM/SVM system previously described is not able to fully exploit the improved generalization capabilities of SVMs due to that SVMs are fed with a segmentation provided by the HMMs. Consequently, the the potential effectiveness of the SVMs is limited by the errors committed in the segmentation stage.

The method suggested in [43] consists in classifying each frame of voice as belonging to a basic class (a phone). Following this approach the need to locate each word in time is avoided and its duration becomes unimportant. In order to go from the classification of each frame to the word chain recognition, The Token Passing algorithm [44] common in HMM-based speech recognition is used. LIBSVM [47] was the software chosen to train the SVMs. The reasons were the following: First, it implements the SMO algorithm [48] that allows a fast SVM training with a fairly high number of samples. And second, it provides an estimated probability value for each frame and candidate phone [49, 50], that will be described later. The main parts of this SVM-based ASR system are described in the following paragraphs.

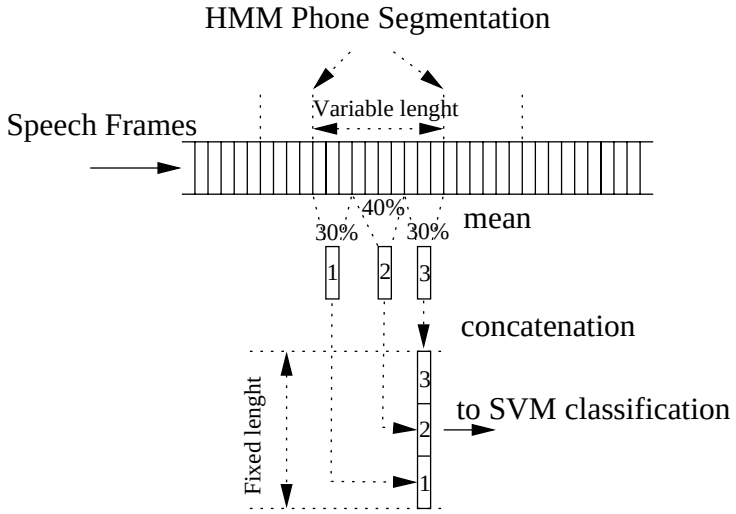

Figure 3: Example of a vector construction for an HMM/SVMs hybrid system.

SVM-based frame by frame classification. Many of the first articles dealing with speech recognition using SVMs mention the possibility of classifying the voice frames directly as a possible method to solve the problem of the variability of duration of speech utterances (different length of the input vectors in the SVM context). This approach was initially rejected because of its high computational cost. Let us make some coarse calculations to gain insight into the problem. Let us consider 31 phones to be identified (typical for Spanish), i.e., a classification problem of 32 classes (the silence is the additional one). If the considered task is a speaker-independent one, the training set should include a high number of speakers: let us consider 100 speakers, though it is a low number. In addition, in order to assure that the phones appear in several contexts and are enough to achieve statistical convergence, we should train with a few minutes of speech from each speaker, for example 10 minutes per speaker. This makes a total of 33.3 hours of voice. If we divided them in frames (computed every 10ms), we would obtain a total of 12.000.000 frames or, in our case, training samples. If we take into account that, in a typical implementation, the entire matrix should be put in memory for training (and that a frame requires, for example, 156 bytes), we would need 1872 GBytes. Furthermore, the CPU time to solve the quadratic problem with so many points should also be considered. At first sight, it seems that this is not a feasible solution. Nevertheless, it is worthwhile to study if the solution is good and, if it was, to worry later about the memory consumption and the computational cost.

In [43] the SVMs are used on a frame by frame basis in order to determine

which class (phone) every frame belongs to. They use as many classes as phones. In particular, for Spanish digits there are 17 phones plus the silence, i.e., every individual voice frame is classified as belonging to one of the 18 classes.

Probability estimations. The SVMs only classify, but they do not give us a reliable measure of the probability of the correctness of the classification. Several ways to estimate this probability can be found in the literature. All of them are based on some kind of mapping between the distances provided by the SVM and the sought probability. The approach followed by LIBSVM considers the actual distances as a measure of "probability". Thus, the posterior probability $S_i(x)$ that a vector x belongs to class i is calculated as

$$S_i(x) = \sum_{\forall j \neq i} f_{ij}(x), \tag{19}$$

where $f_{ij}(x)$ is the distance between the vector x and the hyperplane used to classify between class i and class j. This estimation can be improved using a softmax function as follows:

$$\hat{S}_i(x) = \frac{\exp(S_i(x)/k)}{\sum_j \exp(S_j(x)/k)}, \tag{20}$$

where k is a constant to avoid the function saturation towards 1 or 0.

A more elaborated method makes the assumption that the probability follows a sigmoid function, whose parameters are estimated from the training samples. Thus, the probability p_i that x belongs to class i considering classes i and j can be written as follows [49]:

$$p_i(x) = \frac{1}{1 + \exp(A_{ij} f_{ij}(x) + B_{ij})}, \tag{21}$$

$$p_j(x) = 1 - p_i(x), \tag{22}$$

where in order to avoid severe bias towards the training data, the free parameters, A_{ij} and B_{ij} are estimated on a cross-validation set.

Finally, the conversion of this two-class probability p_{ij} to a multiclass probability P_i is obtained by means of a variation of the Refregier and Vallet method [50].

The Token Passing algorithm [44] transforms a stream of acoustic classifications to a stream of recognized words. Its input is a matrix of probabilities: one row per phone (or subword unit) and one column per frame.

The Token Passing algorithm is an extension of the Viterbi algorithm typically used in continuous speech recognition devised to manage the uncertainty about the number of words in a sentence. Figure 4 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 this 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.

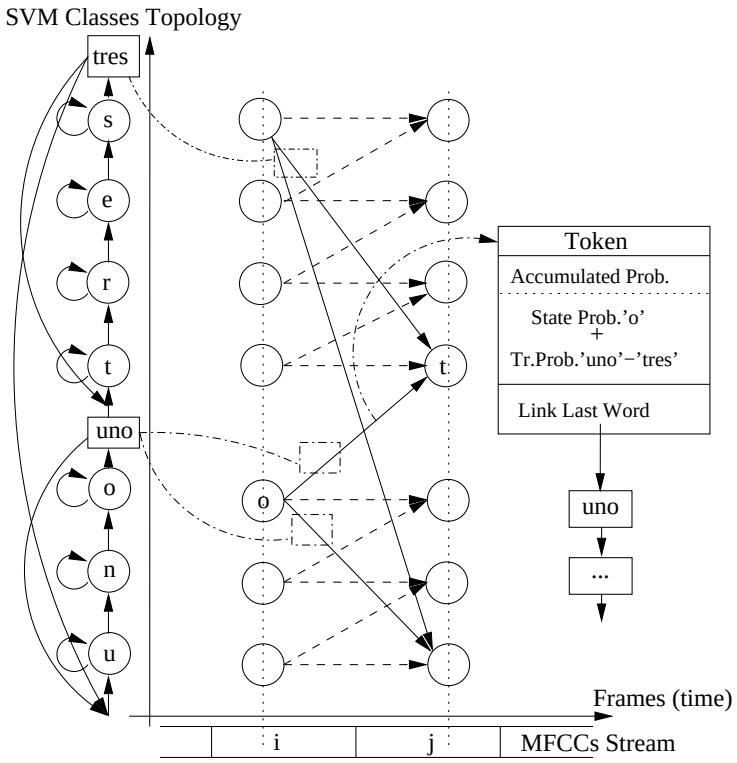

Figure 4: An illustration of the Token Passing Algorithm for a very simple grammar.

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 4). 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.

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, typically, all the Kernels (or a high percentage of them) should be allocated in the computer memory. This 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 Multiclass problem. In order to solve it, the $1-vs-1$ approach is used. This method allows to train all the system using a maximum number of different samples for each class, and to keep limited the use of 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 phones, deciding the winning class by voting.

The definition of classes. When each class is a phone the time variation typically exhibited by actual phones is not taken into account. Some time variation can be embedded through the delta parameters, but better solutions should be considered; for example: 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 to deal with parts of phones.

The last alternative has been chosen because it helps 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, the total number of samples used to train the whole system is effectively increased. 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 the 18 initial classes into $18 \cdot 3 = 54$. Therefore, the number of SVM classifiers to perform the $1-vs-1$ multiclass implementation moves from 153 to 1431.

To 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 mentioned 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.

The results using this approach shows that SVMs can become a competitive alternative to HMMs in continuous speech recognition [43]. With a very small database, 100 utterances, SVMs improve the recognition accuracy of

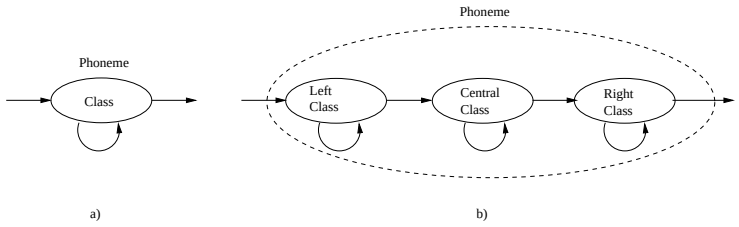

Figure 5: a) Identifying an SVM class per phone; b) Identifying an SVM class as a part of a phone: three SVM classes per phone.

HMMs. Furthermore, they also achieve a similar performance with a large database (100 hours), although at the expense of a huge computational effort. This last result is very encouraging since, due to current training limitations, the SVM-based system only uses the 1.5% of the training database used by HMM-based one.

5 Conclusions and Further Work

In this Chapter we have reviewed the research work dealing with SVMs for ASR. We have started by reviewing the reasons reported in the literature to support the use of SVMs for ASR. Thus, we have explained the characteristics of SVMs that make them valuable from the ASR point of view: first, SVMs are discriminative models, thus more appropriate for classification problems; second, as opposed to ANNs, they have the advantage of being capable to deal with samples of a very higher dimensionality; and third, they exhibit an excellent generalization ability that makes them especially suitable to deal with noisy speech.

After motivating the theme of research, we have described the problems that the attempt to use SVMs in this context has arisen: first, SVMs were originally formulated to process fixed-dimension input vectors and consequently it is not straightforward to manage the speech time variability; second, the SVMs were formerly devised as binary classifiers while the ASR problem is multiclass; and third current SVM training algorithms are not able to manage the huge databases typically used in ASR.

The larger Section of the Chapter is dedicated to present an overview of the solutions that have been proposed to the previously mentioned problems. The exposition have been organized into two subsections, depending of the complexity of the tackled ASR tasks: low- and medium-complexity ASR tasks.

Within the low-complexity tasks subsection, the most relevant alternatives to overcome the problem of speech time variability have been described, highlighting the one based on DTAK-SVMs, that is a genuine SVM-based system able to manage the variable input dimension. The experimental results reveal that the DTAK clearly outperforms the HMM-based system in moderate to highly noisy

environments. In conclusion, we believe that SVMs should be considered as a promising paradigm for the development of robust speech recognition systems. The maximum margin solution provided by SVMs, responsible for their good generalization properties, can be successfully applied to the speech recognition problem.

On the other hand, however, DTAK-SVMs incurs in some performance losses for clean speech or high SNRs. The improvement of the DTAK results for high SNRs remains an open problem for future research: some analysis should be done to gain more insight into the behavior of the DTAK algorithm.

In contrast to low-complexity tasks, the research work is still incipient in the case of medium-complexity ASR tasks. However, the results reported in [43] allow to conclude that the SVMs can be an alternative to the HMMs in continuous speech recognition. On the hand, with a very small database (100 utterances) the SVMs improve the recognition accuracy of HMMs. In addition, similar results are achieved for a large database (100 hours), although at the expense of a huge computational effort. This last result is encouraging since, due to current limitations, the SVM-based system only has used the 1.5% of the training database used by HMM-based one.

There are several proposals to overcome the current difficulties that SVM's algorithms have for handling effectively very large databases [51, 52, 18, 53]. Mega-GSVCs [53], for example, are capable of training classifiers with millions of data while keeping under control the complexity of the resulting machines.

Another way to raise the number of frames that can be used for training is to increase the number of considered classes. For example, different classes could be defined for different phonetic contexts.

Finally, in order to reduce the CPU time consumed in classification, a technique like FC-GSVC [54] or some type of Viterbi pruning could be used.

Acknowledgement

This work has been partially supported by the regional grant (Comunidad Autónoma de Madrid-UC3M) UC3M-TEC-05-059.

References

[1] H. Sakoe, R. Isotani, K. Yoshida, K. Iso, and T. Watanabe. Speaker-Independent Word Recognition using Dynamic Programming Neural Networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 439 – 442, Glasgow, Scotland, 1989.

[2] K. Iso and T. Watanabe. Speaker-Independent Word Recognition using a Neural Prediction Model. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 441–444, Alburquerque, New Mexico (USA), 1990.

[3] J. Tebelskis, A. Waibel, B. Petek, and O. Schmidbauer. Continuous Speech Recognition using Predictive Neural Networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–64, Toronto, Canada, 1991.

[4] H. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*. Boston: Kluwer Academic, Norwell, MA (USA), 1994.

[5] B. Schlkopf and A. Smola. *Learning with kernels*. MIT Press, Cambridge, MA (USA), 2002.

[6] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

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

[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:328–339, 1989.

[9] T. Robinson and F. Fallside. A recurrent error propagation network speech recognition system. *Computer, Speech and Language*, 5:259–274, 1991.

[10] E. Trentin and M. Gori. A survey of hybrid ann/hmm models for automatic speech recognition. *Neurocomputing*, 37:91–126, 2001.

[11] H. Bourlard and N. Morgan. Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks*, 4:893–909, 1993.

[12] T. Robinson, M. Hochberg, and S. Renals. *Automatic Speech and Speaker Recognition - Advanced Topics*, chapter The Use of Recurrent Neural Networks in Continuous Speech Recognition (Chapter 19), pages 159–184. Kluwer Academic Publishers, Norwell, MA (USA), 1995.

[13] W. Reichl and G. Ruske. A hybrid rbf-hmm system for continuous speech recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3335–3338, Detroit, MI (USA), 1995.

[14] D. Ellis, R. Singh, and S. Sivadas. Tandem-acoustic modeling in large-vocabulary recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 517–520, Salt Lake City, Utah (USA), 2001.

[15] B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152, 1992.

[16] F. Pérez-Cruz and O. Bousquet. Kernel Methods and Their Potential Use in Signal Processing. *IEEE Signal Processing Magazine*, 21(3):57–65, 2004.

[17] R. Fletcher. *Practical Methods of Optimization*. Wiley-Interscience, New York, NY (USA), 1987.

[18] A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez, and A.R. Figueiras-Vidal. Weighted Least Squares Training of Support Vector Classifiers leading to Compact and Adaptive Schemes. *IEEE Transactions on Neural Networks*, 12(5):1047–1059, 2001.

[19] S. Fine, J. Navratil, and R.A. Gopinath. A hybrid gmm/svm approach to speaker identification. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 417–420, Salt Lake City, Utah (USA), 2001.

[20] Q. Le and S. Bengio. Client Dependent GMM-SVM Models for Speaker Verification. In *International Conference on Artificial Neural Networks, ICANN/ICONIP,Springer-Verlag*, pages 443–451, 2003.

[21] C. Ma, M.A. Randolph, and J. Drish. A support vector machines-based rejection technique for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 381–384, Salt Lake City, Utah (USA), 2001.

[22] C.W. Hsu and C.J. Lin. A Comparison of Methods for Multi-class Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[23] A. Ganapathiraju, J.E. Hamaker, and J. Picone. Applications of support vector machines to speech recognition. *IEEE Transactions on Signal Processing*, 52:2348–2355, 2004.

[24] N. Thubthong and B. Kijsirikul. Support vector machines for thai phoneme recognition. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9:803–13, 2001.

[25] P. Clarkson and P.J. Moreno. On the use of support vector machines for phonetic classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 585–588, Phoenix, Arizona (USA), 1999.

[26] C. Sekhar, W.F. Lee, K. Takeda, and F. Itakura. Acoustic modelling of subword units using support vector machines. In *Workshop on spoken language processing*, Mumbai, India, 2003.

[27] S. Young. *HTK-Hidden Markov Model Toolkit (ver 2.1)*. Cambridge University, 1995.

[28] J.M. García-Cabellos, C. Peláez-Moreno, A. Gallardo-Antolín, F. Pérez-Cruz, and F. Díaz-de-María. SVM Classifiers for ASR: A Discusion about Parameterization. In *Proceedings of EUSIPCO 2004*, pages 2067–2070, Wien, Austria, 2004.

[29] A. Ech-Cherif, M. Kohili, A. Benyettou, and M. Benyettou. Lagrangian support vector machines for phoneme classification. In *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP '02)*, volume 5, pages 2507–2511, Singapore, 2002.

[30] D. Martín-Iglesias, J. Bernal-Chaves, C. Peláez-Moreno, A. Gallardo-Antolín, and F. Díaz-de-María. *Nonlinear Analyses and Algorithms for Speech Processing*, volume LNAI 3817 of *Lecture Notes in Computer Science*, chapter A Speech Recognizer based on Multiclass SVMs with HMM-Guided Segmentation, pages 256–266. Springer, 2005.

[31] R. Solera-Ureña, D. Martín-Iglesias, A. Gallardo-Antolín, C. Peláez-Moreno, and F. Díaz-de-María. Robust ASR using Support Vector Machines. *Speech Communication, Elsevier (submitted)*, 2006.

[32] S.V. Gangashetty, C. Sekhar, and B. Yegnanarayana. Combining evidence from multiple classifiers for recognition of consonant-vowel units of speech in multiple languages. In *Proceedings of the International Conference on Intelligent Sensing and Information Processing*, pages 387–391, Chennai, India, 2005.

[33] H. Shimodaira, K.I. Noma, M. Nakai, and S. Sagayama. Support vector machine with dynamic time-alignment kernel for speech recognition. In *Proceedings of Eurospeech*, pages 1841–1844, Aalborg, Denmark, 2001.

[34] H. Shimodaira, K. Noma, and M. Nakai. *Advances in Neural Information Processing Systems 14*, volume 2, chapter Dynamic Time-Alignment Kernel in Support Vector Machine, pages 921–928. MIT Press, Cambridge, MA (USA), 2002.

[35] L. R. Rabiner, A.E. Rosenberg, and S.E. Levinson. Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(6):575–582, 1978.

[36] J. R. Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17:137–152, 2003.

[37] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. Technical report, Dept. of Computer Science, Univ. of California, 1998.

[38] N.D. Smith and M.J.F. Gales. Using SVMs and discriminative models for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 77–80, Orlando, Florida (USA), 2002.

[39] N.D. Smith and M.J.F. Gales. *Advances in Neural Information Processing Systems 14*, volume 14, chapter Speech recognition using SVMs, pages 1197–1204. MIT Press, Cambridge, MA (USA), 2002.

[40] N.D. Smith and M. Niranjan. Data-dependent Kernels in SVM Classification of Speech Patterns. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, volume 1, pages 297–300, Beijing, China, 2000.

[41] V. Wan and S. Renals. Speaker verification using sequence discriminant support vector machines. *IEEE Transactions on Speech and Audio Processing*, 13:203–210, 2005.

[42] A. Ganapathiraju, J. Hamaker, and J. Picone. Hybrid SVM/HMM Architectures for Speech Recognition. In *Proceedings of the 2000 Speech Transcription Workshop*, volume 4, pages 504–507, Maryland (USA), May 2000.

[43] J. Padrell-Sendra, D. Martín-Iglesias, and F. Díaz-de-María. Support vector machines for continuous speech recognition. In *Proceedings of the 14th European Signal Processing Conference*, Florence, Italy, 2006.

[44] S. J. Young, N. H. Russell, and J. H. S. Thornton. Token Passing: a Conceptual Model for Connected Speech Recognition Systems. Technical report, CUED Cambridge University, 1989.

[45] Piero Cosi. Hybrid HMM-NN architectures for connected digit recognition. In *Proceedings of the International Joint Conference on Neural Networks*, volume 5, pages 85–90, 2000.

[46] A. Juneja and C. Espy-Wilson. Segmentation of continuous speech using acoustic-phonetic parameters and statistical learning. In *Proceedings of the 9th International Conference on Neural Information Processing, (ICONIP '02)*, volume 2, pages 726–730, 2002.

[47] Ch. Chih-Chung and L. Chih-Jen. *LIBSVM: a library for support vector machines*, 2004.

[48] J. C. Platt. *Advances in Kernel Methods: Support Vector Learning*, chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, Cambridge, MA (USA), 1999.

[49] J. C. Platt. *Advances in Large Margin Classifiers*, chapter Probabilities for SV Machines, pages 61–74. MIT Press, 1999.

[50] T. F. Wu, C. J. Lin, and R. C. Weng. Probability estimates for multiclass classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, 2004.

[51] C.J.C. Burges. Simplified support vector decision rules. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 71–77, Bari, Italy, 1996.

[52] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, Amelia Island, Florida (USA), 1997.

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

[54] E. Parrado, J. Arenas, I. Mora, A. Figueiras, and A. Navia. Growing Support Vector Classifiers with Controlled Complexity. *Pattern Recognition*, 36:1479–1488, 2003.