Advances in the Application of Support Vector Machines as Probabilistic Estimators for Continuous Automatic Speech Recognition

by

Daniel Bolanõs Alonso, Licenciado en Ingeniería Informática

Dissertation

Dissertation submitted to the

Departamento de Ingeniería Informática, Universidad Autónoma de Madrid

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Departamento de Ingeniería Informática, Universidad Autónoma de Madrid

November 2008

Escuela Politécnica Superior

Department:	Ingeniería Informática. Escuela Politécnica Superior (EPS). Universidad Autónoma de Madrid (UAM).
Thesis title:	Advances in the Application of Support Vector Machines as Probabilistic Estimators for Continuous Automatic Speech Recognition
Author:	Daniel Bolaños Alonso
Supervisor:	Wayne H. Ward Ph.D. Quantitative Psychology, University of Colorado (CU), USA Research Professor (CU), USA
Date:	November, 2008.
Committee:	Luis Alfonso Hernández Gómez Universidad Politécnica de Madrid, Spain
	Joaquín González Rodríguez Universidad Autónoma de Madrid, Spain
	Ronald A. Cole University of Colorado at Boulder, USA
	Ascensión Gallardo-Antolín Universidad Carlos III de Madrid, Spain
	Doroteo Torre Toledano Universidad Autónoma de Madrid, Spain

To my family

Abstract

In recent years, the number of applications that benefit from speech processing techniques has grown significantly. This growth has been motivated fundamentally by two factors, the creation of new speech-enabled services and the enhancement of speech processing techniques. Nevertheless, while the recognition accuracy of state-of-the-art systems is satisfactory for a number of real-world applications, the reduction of speech recognition error rates, especially in large vocabulary continuous speech recognition systems, still stands as a major challenge.

Nowadays, a generative modeling approach based on the combination of Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) has become a de facto standard for speech recognition. HMMs allow to model the temporal dynamics of speech and GMMs are used to model the probability of the observations (feature vectors extracted from the speech). The parameters of HMMs can be estimated following a number of different criteria. While originally Maximum Likelihood Estimation (MLE) was the most widely adopted criterion, the fact that it relies exclusively on a Empirical Risk Minimization (i.e. minimizing the errors on the training data) has caused the flourishing of a variety of discriminative training criteria that allow a better generalization of the models and a closer solution to the word error rate minimization problem.

At this point, one may argue: why bother learning the underlying distribution of the data (i.e the probability of the observations) using generative models and then training them using a discriminative training criterion if it is possible to learn directly the boundaries between samples belonging to different classes using discriminative classifiers? This question has been formulated

many times in the literature and some discriminative classifiers, being Artificial Neural Networks the most prominent one, have been employed to build automatic speech recognition (ASR) hybrid-systems that perform similarly to GMM/HMM state-of-the-art ones. However, a number of issues have prevented those classifiers to replace the GMM/HMM paradigm as the standard modeling approach for (ASR). Among other issues, traditional discriminative classifiers suffer from the same lack of generalization ability that HMMs trained under MLE do, i.e. perform only an Empirical Risk Minimization what typically results in poor generalization performance.

In the other hand, Support Vector Machines (SVMs) present clear advantages over traditional discriminative techniques. SVMs are a relatively recent supervised learning technique based on recent advances in statistical learning theory. SVMs belong to the family of large-margin classifiers and their main characteristic is that they simultaneously minimize the empirical classification error (Empirical Risk Minimization) and maximize the geometric margin between samples of different classes (Structural Risk Minimization). This Structural Risk Minimization principle is based on controlling the tradeoff between the complexity of the decision function (capacity of the model) and the classification errors on the training data. For this reason, SVMs are a very successful discriminative approach that have been effectively used in a number of pattern recognition tasks.

In the case of speech processing, SVMs have shown superior performance compared to alternative discriminative techniques in a number of classification tasks like speaker verification or phonetic classification. However, the potential of SVMs to be applied to continuous ASR, where both classification and segmentation of lexical units need to be addressed simultaneously, is still unclear. This thesis work is focused on exploring the applicability of SVMs to continuous speech recognition in a stand-alone architecture. In this thesis, a method for utilizing SVMs as probabilistic estimators of emission probabilities in a continuous ASR system is presented and evaluated. While the utilization of SVMs for binary classification tasks is straightforward, in the case of ASR, a large number of classes is involved and training on millions of feature vectors is required to attain satisfactory recognition accuracy. In this thesis work a SVM/HMM speech decoding system is proposed and implemented in which SVMs are used as emission probability estimators. Experimental results show that the proposed system outperforms a comparable GMM/HMM decoder in terms of word accuracy. However, this system still presents several disadvantages concerning the number

of parameters and scalability. In this respect several techniques are proposed in order to minimize those disadvantages. In one hand an algorithm is introduced that significantly reduces the number of classifiers that need to be carried out during decoding with no impact in the recognition accuracy. In the other hand, techniques for reusing kernel evaluations across classifiers during decoding and for reducing the storage requirements of the acoustic models are proposed and evaluated. Additionally, it has been experimentally shown that SVM solvers recently proposed in the literature exhibit training time asymptotically linear with the number of samples in the task of feature frames classification. This is particularly interesting not only for scaling the proposed SVM/HMM system to larger datasets but for a broad range of speech processing tasks in which SVMs are trained on millions of samples.

Abstract (Spanish)

En los últimos anõs, el número de aplicaciones que se benefician de técnicas de procesado del habla ha crecido significativamente. Este crecimiento ha estado motivado fundamentalmente por dos factores, la creación de nuevos servicios accesibles mediante la voz y el perfeccionamiento de las técnicas de procesado del habla. Sin embargo, mientras que la precisión de reconocimiento de sistemas del estado-del-arte es satisfactoria para numerosas aplicaciones en el mundo real, la reducción de las tasas de error de reconocimiento, especialmente en sistemas de habla continua y gran vocabulario aún permanece como un gran desafío.

Actualmente, un procedimiento de modelado generativo basado en la combinación de Modelos Ocultos de Markov (MOM) y Modelos de Mezcla de Gausianas (MMG) se ha convertido en el estándar de facto en reconocimiento de voz. Los MOM permiten modelar la dinámica temporal de la voz mientras que los MMG son usados para modelar la probabilidad de las observaciones (vectores de características extraidos del habla). Los parámetros de los MOM pueden ser estimados siguiendo numerosos criterios diferentes. Mientras que originalmente la Estimación basada en Máxima Probabilidad (EMP) fue el criterio más ampliamente adoptado, el hecho de que se basa exclusivamente en una Minimización del Error Empírico (MEE) (es decir, minimizar los errores en los datos de entrenamiento) ha causado el florecimiento de una variedad de criterios de entrenamiento discriminativo que permiten una mejor generalización de los modelos y una solución mas cercana al problema de minimización de la tasa de error.

En este punto, se podría argumentar: ¿Por qué molestarse en aprender la distribución subyacente de los datos (es decir, la probabilidad de las observaciones) utilizando modelos generativos y luego entrenarlos utilizando un criterio de modelado discriminativo si es posible aprender directamente los limites entre muestras pertenecientes a clases diferentes? Esta pregunta ha sido formulada muchas veces en la literatura y varios clasificadores discriminativos, siendo las Redes Neuronales Artificiales (RNA) el más prominente, han sido empleados para construir sistemas de reconocimiento automático del habla (RAH) que rinden de manera similar a sistemas MMG/MOM. Sin embargo, numerosas cuestiones han evitado que esos clasificadores reemplacen al paradigma MMG/MOM como el estandard de modelado en sistemas de RAH. Entre otras cuestiones, los clasificadores discriminativos tradicionales sufren de la misma falta de habilidad de generalización que tienen los MOM entrenados bajo EMP, es decir, producen sólo una Minimización del Error Empírico, lo que típicamente resulta en una pobre generalización.

Por otro lado, las Maquinas de Soporte Vectorial (MSV) presentan claras ventajas respecto a los clasificadores discriminativos tradicionales. Las MSV son una técnica reciente de aprendizaje supervisado basada en avances recientes en la teoría del aprendizaje estadístico. Las MSV pertenecen a la familia de clasificadores de gran-margen y su principal característica es que minimizan simultáneamente el error de clasificación empírico (Minimización del Riesgo Empírico) y maximizan el margen geométrico entre muestras de diferentes clases (Minimización del Riesgo Estructural). La Minimización del Riesgo Estructural se basa en controlar el compromiso entre la complejidad de la función de decisión (capacidad del modelo) y los errores de clasificación en los datos de entrenamiento. Por esta razón, las MSV son un procedimiento discriminativo de gran éxito que ha sido utilizado en numerosas tareas de reconocimiento de patrones.

En el caso del procesado del habla, las MSV han mostrado un rendimiento superior comparado con técnicas discriminativas alternativas en tareas de clasificación como verificación de locutor y clasificación fonética. Sin embargo, el potencial de las MSV para ser aplicadas a reconocimiento automático de habla continua, donde la clasificación y segmentación de unidades léxicas necesita realizarse simultáneamente, no esta claro. Este trabajo de tesis esta enfocado a explorar la aplicabilidad de las MSV al reconocimiento de habla continua en una arquitectura independiente. En esta tesis será presentado un método para utilizar las MSV como estimadores

probabilísticos de las probabilidades de emisión en un sistema de reconocimiento de habla continua. Mientras que la utilización de las MSV en tareas de clasificación binaria es simple y directa, en el caso del RAH hay un gran número de clases involucradas y es necesario entrenar en millones de vectores de características para lograr una precisión de reconocimiento satisfactoria. En este trabajo de tesis se ha propuesto e implementado un sistema de reconocimiento del habla bajo el paradigma MSV/MOM en el que las MSV son utilizadas como estimadores de las probabilidades de emisión. Resultados experimentales muestran que el sistema propuesto supera a un sistema MMG/MOM comparable en terminos de precisión de reconocimiento. Sin embargo, este sistema presenta varias desventajas relacionadas con el número de parámetros y la escalabilidad. A este respecto, se han propuesto varias técnicas para minimizar dichas desventajas. Por un lado se ha introducido un algoritmo que reduce significativamente el número de clasificadores que han de evaluarse durante la decodificación sin que se haya causado impacto alguno en la precisión del reconocimiento. Por otro lado, se han propuesto y evaluado varias técnicas para reutilizar evaluaciones del kernel entre clasificadores durante la decodificación y para reducir los requerimientos de almacenamiento de los modelos acústicos. Adicionalmente, se ha mostrado experimentalmente que tecnicas de resolución de MSV recientemente propuestas en la literatura, exhiben tiempo de entrenamiento asintóticamente lineal con el número de muestras para la tarea de clasificación de vectores de características. Esto es particularmente interesante no sólo para escalar el sistema MSV/MOM propuesto a conjuntos de datos más grandes, sino para un amplio rango de tareas de procesado del habla en las que las MSV son entrenadas con millones de muestras.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

Speech processing technology started more than fifty years ago. Since then, the field has broaden enormously and numerous techniques have been developed in a wide variety of speech related topics. For example: speech coding, speech classification, speech recognition, speech synthesis, speaker verification, etc. Among those, continuous speech recognition is probably the topic that has received the greatest attention. This topic relates to the extraction of the textual content from speech segments as they are uttered by the speaker.

Improving the quality of an automatic speech recognition system is not only a goal by itself but a strong requirement for improving the quality of a great number of speech-enabled systems like automatic inquiry systems and dialog systems. In such systems the goal is to thoroughly understand and process the user request in the minimum number of turns. An accurate and reliable text transcription of the speech query helps in the reduction of verification and repetition turns, thus accelerating the interaction with the system and improving the user experience.

Nowadays, most state-of-the art continuous ASR systems make use of Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) for the acoustic modeling. In this generative approach, GMMs model the distribution of the input data (observations at each of the states of the HMMs) and HMMs model the temporal dynamics of speech. HMMs are especially suitable for continuous speech recognition because, thanks to the independence assumption between adjacent feature frames, allow the application of dynamic programming techniques in which the computation of the most likely sequence of states at time t is based on the most likely sequence of states at time $t-$

1 and the feature vector observed at time t. Originally, the Maximum Likelihood Estimation (MLE) was the most widely adopted criterion for training the HMMs parameters, however, nowadays a wide variety of discriminative training criteria have replaced MLE. The discriminative training of HMM parameters allows a better generalization of the models and a closer solution to the word error rate minimization problem.

In the other hand, machine learning has made an enormous progress during the last decades. One of the major results is the development of the Structural Risk Minimization (SRM) principle, that allows a tradeoff between the complexity of the decision function (used to separate samples from different classes) and the classification errors in the training data. This SRM principle plays a fundamental role in the generalization capacity of the model and, thus, in the ability to predict unseen samples. A proof of this excellent generalization ability is the vast number of pattern recognition tasks that currently benefit from the use of Support Vector Machines (SVMs).

In the case of speech processing, SVMs have been successfully applied to a number of tasks like phonetic classification, speaker verification, hypothesis rescoring, etc. However, the application of SVMs to continuous speech recognition in a stand-alone system has not been fully explored yet. While continuous speech recognition can be roughly viewed as a classification of variable-length speech units, the fact that the starting and ending times of the lexical units under classification are unknown, increases considerably the complexity of the problem. On top of that, speech recognition involves a large number of different classes and training on very large datasets, which are some of the weaknesses of SVMs. Nonetheless, the fact that SVMs are by nature a discriminative technique (in contrast with the generative GMM/HMM approach) and their excellent generalization performance (what is of the utmost importance in speech recognition) makes this technique very appealing from the point of view of speech recognition.

This thesis is dedicated to explore the utilization of SVMs as probabilistic estimators of emission probabilities for continuous speech recognition. The intention is to propose a stand-alone SVM-based speech recognition system, to implement it and identify its weaknesses. Additionally the recognition accuracy of the system will be compared to that of a conventional GMM/HMM system.

1.1 Thesis goals

- To explore the applicability of SVMs as probabilistic estimators for continuous automatic speech recognition. This includes the selection of an adequate model topology and training methodology as well as an study of the decoding framework.

- To implement a fully functional SVM-based continuous speech recognizer and compare its performance to that of a conventional GMM-based one. The performance will be measured in terms of word error rate. The comparison will be made on a very challenging corpora of children's speech.

- To identify the advantages and disadvantages of the proposed system and implement techniques to address its weaknesses. This includes analyzing scalability issues related to the training on very large datasets (millions of samples), which is a well known shortcoming of SVMs.

- To study a technique for incorporating context-dependency into the proposed system. The technique should be be based on a top-down strategy that allows the clustering of unseen triphones.

- As an additional goal, and connected to the main topic of this thesis (SVM-based ASR) by the use of SVMs as probabilistic estimators. It will be explored the application of SVM-based features extracted from phone graphs to the task of pronunciation scoring. Pronunciation scoring is the task of labeling word realizations as correctly/incorrectly pronounced.

1.2 Thesis overview

Chapter 2 of this thesis is devoted to, firstly, introduce the mathematical foundations of Support Vector Machines and, secondly, discuss their utilization as probabilistic estimators and strategies for dealing with multiclass classification tasks. The chapter is mainly a summarization of the state-of-the-art in the application of SVMs to pattern recognition and probability estimation. However, in addition to that, existing strategies are compared from different perspectives and their suitability

for a particular task or another is analyzed. The intention is to make an entry point for the rest of the chapters that will be focused in the application of SVMs to a particular task: speech processing. This chapter will be referenced back from other chapters in this thesis when motivating some of the decisions made.

Chapter 3 is dedicated to introduce the utilization of SVMs to speech processing. It starts by making a comparison between SVMs and other discriminative methods (ANNs) that already showed some success applied to speech processing tasks. Afterward, the problem of modeling the temporal structure of speech is analyzed from the perspective of SVMs. Different techniques are analyzed along with their advantages/disadvantages when applied to speech processing tasks of different nature including continuous speech recognition. Finally, a novel application on the utilization of SVM-based features for pronunciation scoring is presented.

In chapter 4 the application of SVMs as probabilistic estimators for continuous speech recognition is explored. A stand-alone SVM/HMM based speech recognition system is proposed and experimentally evaluated. The system makes use of SVMs as probabilistic estimators of emission probabilities while sill utilizing HMMs to model the temporal dynamics of speech as in conventional GMM/HMM systems.

Chapter 5 of this thesis addresses the problem of training pairwise SVM classifiers to separate feature vectors of phonetically close classes.

Chapter 6 is devoted to explore some ideas on the incorporation of context-dependent modeling into the proposed SVM/HMM system. A technique is proposed that finds clusters of triphones which pairwise separability is maximized.

Chapter 7 is dedicated to address scalability issues that have been identified in the proposed SVM/HMM system. Several techniques are introduced in order to reduce the training time, the decoding time and the memory requirements of the acoustic models.

Finally, chapter 8 is dedicated to present the conclusions and major contributions of this thesis work. In addition to that some lines of future research are proposed.

Chapter 2

Support Vector Machines

Support Vector Machines (SVMs) [1] are a relatively recent supervised learning technique based on recent advances in statistical learning theory. SVMs belong to the family of large-margin classifiers and their main characteristic is that they simultaneously minimize the empirical classification error and maximize the geometric margin between samples of different classes. One of the most appealing properties of SVMs is that besides performing empirical risk minimization, they carry out a structural risk minimization by controlling the tradeoff between the complexity of the decision function (capacity of the model) and the classification errors on the training data.

This chapter is dedicated to, first, outline the mathematical foundations of this technique and, second, discuss several existing methods toward its application to multiclass-classification tasks and probabilities estimation. The intent is to make a short overview of the state-of-the-art on these issues, that will serve as the basis for the next chapters in which SVMs will be explored from the point of view of speech processing, and particularly for continuous speech recognition.

2.1 Mathematical Foundations

SVMs are binary classifiers belonging to the group of large-margin classifiers, which have proven to be effective in delivering high predictive accuracy. Given a training set composed of labeled vectors $\{\mathbf{x}_i, y_i\}, i = 1, \ldots, l, \mathbf{x}_i \in \mathbf{R}^d$ are the d-dimensional feature vectors and $y_i \in \{1, -1\}$ are the labels. A SVM finds (when existing, i.e. in the linearly separable case) the separating hyperplane that

maximizes the margin between the samples belonging to both classes $\{1, -1\}$. This hyperplane not only guarantees the best classification of the training samples but offers the maximum generalization ability for unseen samples.

2.1.1 Linear Support Vector Machines

The linearly separable case

First, it will be considered the linearly separable case, i.e. there exists a separating hyperplane that divides the feature space into two regions, each of both containing the training samples of one class. If that hyperplane exists the following inequalities must be satisfied:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{if } y_i = +1 \tag{2.1}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{if } y_i = -1 \tag{2.2}$$

Where $\mathbf{w}$ is normal to the hyperplane and $\frac{|b|}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin. Inequalities 2.1 and 2.2 can be combined into the following expression:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{2.3}$$

Considering the points for which equalities in 2.1 and 2.2 hold, it is possible to define two hyperplanes $\mathcal{H}_1$ and $\mathcal{H}_2$ that are parallel (both have $\mathbf{w}$ as normal) and no training points fall between them. Given that the perpendicular distances of $\mathcal{H}_1$ and $\mathcal{H}_2$ to the origin are $\frac{|1-b|}{\|\mathbf{w}\|}$ and $\frac{|-1-b|}{\|\mathbf{w}\|}$ respectively, the distance between these hyperplanes and the largest margin hyperplane is $\frac{1}{\|\mathbf{w}\|}$. Thus, the margin is $\frac{2}{\|\mathbf{w}\|}$ and it can be maximized by minimizing $\|\mathbf{w}\|^2$. An interesting observation is that, once the optimal hyperplanes $\mathcal{H}_1$ and $\mathcal{H}_2$ are found, the training points contained in them are called support vectors and any other training points have no effect in the selection of these hyperplanes.

A convenient representation method for the constraints present in 2.3 is the Lagrange multipliers. Using a positive Lagrange multiplier $\alpha_i, i = 1, \ldots, l$ for each inequality constrain the

following Lagrangian is obtained:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{2.4}$$

Minimizing L_P with respect to $\mathbf{w}$ and b consists of solving a quadratic programming problem, which represents the goal of the training. When the number of training samples is large enough, solving this quadratic problem directly can be computationally extremely expensive so many different techniques have been proposed in the literature to find an approximate solution to this problem[1]. This optimization problem can be converted into the dual form 2.5 subject to the conditions given by 2.6 and 2.7. In this case the optimization problem consists of maximizing the dual problem L_D instead of minimizing the primal problem L_P.

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{y}_i \tag{2.5}$$

$$0 \leq \alpha_i \tag{2.6}$$

$$\sum_i \alpha_i y_i = 0 \tag{2.7}$$

The dual form, in this case, simplifies the optimization problem by simplifying the constraints. However, the major point for the dual formulation is that the training points only appear in the form of dot products thus allowing the application of the kernel-trick when dealing with nonlinear decision functions (see section 2.1.2).

Once the training of the SVM is complete, it is possible to classify an unseen input sample by determining in which side of the largest margin hyperplane obtained the sample lies. Considering that the possible classes are $y \in \{-1, 1\}$, the class label can be obtained as expressed in 2.8.

$$y = sgn(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.8}$$

[1]Chapter 7 is dedicated to outline some of those techniques and to explore one of the most promising ones in the case of a speech-processing classification task.

The non linearly separable case

Now it will be considered the case when it is not possible to separate the data linearly, i.e. there exist no hyperplane in the feature space that is able to separate all the positive examples from the negative ones. In this case the quadratic problem has no solution and maximizing the objective function 2.5 will result in arbitrarily large values. In order to cope with this problem, the constraints expressed in 2.1 and 2.2 can be relaxed and reformulated as follows:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \;\text{ if }\; y_i = +1 \tag{2.9}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \;\text{ if }\; y_i = -1 \tag{2.10}$$

$$\xi_i \geq 0 \;\; \forall i \tag{2.11}$$

Accordingly, an extra cost for errors is introduced by changing the objective function to be minimized from $\frac{\|\mathbf{w}\|}{2}$ to $\frac{\|\mathbf{w}\|}{2} + C(\sum_i \xi_i)^k$ where C is a regularization parameter selected by the user (commonly referred as the error penalty or the cost) that allows to control the amount of overlap by assigning different penalty to errors. Selecting $k = 1$ neither the ξ_i, nor their Lagrange multipliers appear in the dual problem, which makes the expression to maximize simpler:

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{y}_i \tag{2.12}$$

subject to the following conditions:

$$0 \leq \alpha_i \leq C \tag{2.13}$$

$$\sum_i \alpha_i y_i = 0 \tag{2.14}$$

Note that the only difference with respect to the case when the optimal separating hyperplane exists, is that the α_i now have an upper bound of C. This parameter is used to penalize classification errors, so a high value of C reduces the number of misclassified training samples by fitting the decision boundary to the training samples. Hence, a too high value of C may produce irregular decision boundaries and it is prone to produce overfitting, which deteriorates the generalization ability of

the classifier. In the other hand a too small value of C may make the training algorithm to find a separating hyperplance with a too large margin, which may not produce satisfactory classification results. The optimal value of C depends on the application, and it is typically estimated doing cross-validation on the training data.

2.1.2 Nonlinear Support Vector Machines

For some separating tasks a decision function that is a linear function of the data may not be able to make a satisfactory separation of the data. In these cases it is of particular interest using a decision function that is not a linear function of the data. SVMs using this kind of decision functions are called nonlinear SVMs and are based on the kernel-trick [2]. As it will be shown next, the essence of the idea lies in the fact that the training samples $\mathbf{x}_i$ only appear in the training equations in the form of dot products $\mathbf{x}_i \cdot \mathbf{x}_j$ (see equation 2.5). The first step consists of mapping the training data from the feature space $\mathbf{R}^d$, in which they live, to another (possibly infinite dimensional) Euclidean space $\mathcal{H}$ using the mapping Φ as expressed in 2.15.

$$\Phi : \mathbf{R}^d \mapsto \mathcal{H} \tag{2.15}$$

Now, having a kernel function in the form of $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ it is possible to replace the dot products $\mathbf{x}_i \cdot \mathbf{x}_j$ in the training equations by $K(\mathbf{x}_i, \mathbf{x}_j)$ thus avoiding the explicit computation of $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ that lives in the potencially inifinite dimensional space $\mathcal{H}$. Note that the kernels $K(\mathbf{x}_i, \mathbf{x}_j)$ that can be used must fulfill the Mercer's condition [1].

Some of the most popular non linear kernels are the following:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p \tag{2.16}$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2} \tag{2.17}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta) \tag{2.18}$$

Where 2.16 is a polynomial kernel of degree p, 2.17 is a Gaussian Radial Basis Function (RBF) and 2.18 is the sigmoid kernel. These kernels present parameters which optimal value depends on

the task and, thus, are typically estimated doing cross-validation on the training data. One of the most appealing properties of kernel-based learning methods is that kernels are used as a similarity measure between training samples and, hence, can be used to incorporate prior-knowledge from the domain[2] when it is available. However, in some cases this can be view as a shortcoming, given that sometimes, specifying a suitable kernel for an application may not be straightforward.

The decision function of a non-linear SVM, used for classifying new samples, is expressed in 2.19 and it is equivalent to expression 2.8 with the sole exception of replacing the dot product between the input sample $\mathbf{x}$ and the support vectors s_i by a kernel evaluation $K(\mathbf{s}_i, \mathbf{x})$.

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b\right) \tag{2.19}$$

More details on the utilization of these kernels and a full review of SVMs and related techniques can be found in [4].

2.2 Support Vector Machines as probabilistic estimators

When SVMs are used for classification tasks, the process of labeling an unseen input sample $\mathbf{x} \in \mathbf{X}$ as belonging to the class $y = \{-1, 1\}$ is carried out using the expression 2.8. While for many applications it is enough with having a prediction of the sample's class, for some other applications it is also desirable to have a class-posterior probability value $p(y = +1|\mathbf{x})$ indicating the likelihood of the sample to belong to the class. A classifier that produces probabilistic values can not only be used for standard classification but as a statistical estimator.

The output of a SVM is a distance measure between a test pattern and the decision boundary resulting from the training. Thus, the relationship between this uncalibrated value and the posterior class probability $p(y = +1|\mathbf{x})$ that the pattern $\mathbf{x}$ belongs to the class $y = +1$ is not clear. One mechanism to cope with this problem consists of mapping the margin or distance SVMs produce to a class-posterior probability. This can be carried out using a sigmoid [5] as expressed in 2.20),

[2]For example, in [3] the so-called *even-polynomial* kernel is introduced for classifying acoustic waveforms. This kernel takes into account the fact that a speech waveform and its inverted version are perceived as being the same. In [3], it is reported that a classification accuracy improvement can be achieved by incorporating this prior knowledge into the kernel function.

where the parameters A and B need to be estimated by cross-validation on the training data.

$$p(y = +1|\mathbf{x}) = g(f(\mathbf{x}), A, B) \equiv \frac{1}{1 + e^{Af(\mathbf{x})+B}} \tag{2.20}$$

Alternative methods for obtaining probability estimates from SVMs can be found in [6].

2.3 Multiclass classification

SVM classifiers are intended for binary classification so for K-class ($K > 2$) separation tasks[3], several strategies have been proposed in the literature. Attending to how the optimization formulation is carried out, these strategies can be broadly divided in two categories:

- Considering all the classes at once and doing only one optimization formulation. Some of the most popular techniques falling into this category are [7] and [8].
- Decomposing the multiclass classification problem into series of binary classification problems for which a different optimization formulation is carried out. Among these techniques are the well known one-versus-one and one-versus-rest techniques.

In practice, it has been found that techniques falling into the first category, although more elegant, may lead to a very slow convergence resulting in very long training time. The reason is that the optimal value for the cost parameter C is typically high. Details on this can be found on [9], where a comparison between several multiclass classification techniques is done. For this reason, and given that decomposition techniques are by far the most widespread, the rest of this section will be entirely centered on them.

Decomposition techniques address two linked problems, firstly, decomposing the multiclass classification problem into a series of binary classification subproblems for which a SVM classifier is trained and, secondly, combining the output of those classifiers to make the final call. Both decomposition and combination methods provide an answer to different questions:

[3] A vast number of real-world classification tasks indeed involve more than two classes. In the particular case of speech processing, speaker identification or phonetic classification are two clear examples of tasks where multiclass classification is required.

- Decomposition: what is the most effective mechanism for grouping the training samples of different classes to form the positive and negative subset of samples used for training each SVM classifier?
- Combination: once the different classifiers have been trained, how can the output of these classifiers be combined to make the final call of whether an unseen sample belongs to a class or another?

In this section, the most widespread decomposition strategies for multi-class classification using SVMs will be reviewed. For the sake of clarity these strategies are grouped attending to both questions raised before. Note that, while the decomposition techniques are applied at the training-stage, the combination techniques are applied in the classification-stage. Thus, for a given decomposition technique, as it will be shown later on, usually several combination techniques may be applied.

2.3.1 Decomposition methods for multiclass classification

One versus rest

According to the one-versus-rest strategy [10] one classifier is trained for each class k_i to discriminate it from the other classes. The training samples available for that class become the positive samples while the negative samples are all the training samples belonging to the rest of the classes. This method is characterized by the following properties:

- Reduced number of classifiers: the number of classifiers that need to be trained grows linearly with the number of classes K.
- Large training sets: all the training samples are used for the training of each SVM. Hence, even though the number of available samples for each class is not too high, when the number of classes K is considerable, training on very large datasets is often required.
- Classifiers are trained on very unbalanced datasets: independently of how the training data is distributed over the different classes, the one-versus-rest approach always produces very unbalanced training sets. The reason is that the number of negative samples used to train any

classifier is in average significantly higher than the number of positive ones. For example, in the case of phonetic classification in the English language, it is clear that there are significantly fewer training samples available for any given phonetic class than for the remaining ones. However, in order to train on balanced datasets it is possible to carry out a sample selection procedure on the negative set of samples with the intention of using as negative evidence only the subsets of negative samples that are more informative. The problem is that it is not clear how to determine an effective selection criterion.

One versus one

In the one-versus-one approach [11] one classifier is trained to separate each possible pair of classes (k_i, k_j), being the positive training samples those of k_i and the negative samples those belonging to k_j. Hence, if the total number of classes is K, the number of classifiers that need to be trained is $K(K-1)/2$. This decomposition strategy presents the following properties:

- Considerable number of classifiers: the number of pairwise classifiers that need to be trained, $(K(K-1)/2)$, grows quadratically with the number of classes K.
- Relatively reduced training sets: each classifier is trained using only training samples belonging to a pair of classes, being the rest of the samples excluded from the training of that classifier.
- Classifiers are trained on relatively balanced datasets: the degree of how balanced the datasets are is relative to the distribution of the training samples across the different classes and is independent of the total number of classes K.

Part versus part

This strategy, proposed in [12], goes one step further than the one-versus-one strategy by splitting the two-class binary problems into smaller binary subproblems. The training samples belonging to each of the classes involved in a binary classification problem are split into a number of disjoint subsets (or clusters) which size is selected based on two criteria:

- Number of samples of each class must be balanced. Thus, splitting a binary problem into smaller subproblems will result in a larger number of subsets of the class for which more training samples are available.

- Computational power available: the more limited the resources the smaller is the upper bound of the number of samples that are taken from each class to train a classifier.

In [12], training samples belonging to a class are assigned randomly to different clusters. However, a clustering procedure based on samples similarity or prior knowledge (when available) may potentially produce more homogeneous clusters at the expense of preprocessing time. This part-versus-part decomposition strategy presents the following properties:

- Great number of classifiers (with respect to other strategies), which depends on the stopping criterion used for the splitting.

- Classifiers are trained on very balanced and small datasets: this is clearly a desirable feature. One of the interesting properties of a balanced dataset (other than the fact that the decision function resulting from the training typically generalizes equally well for samples of both classes) is that it allows an efficient parallelization of the training processes when multiple processors are available.

- It is not clear how to determine the stopping criterion for the splitting[4], which may depend on the task.

Once all the required SVMs are trained, they can be combined to make the final call using a min-max modular (M^3) SVM.

2.3.2 Combination of classifiers

In the previous section, several strategies have been described that can be utilized to select the set of classifiers that are trained for separating K classes in a multiclass classification task. In this section, the most widespread methods for combining the output of those classifiers to label an unseen sample are described.

[4]It can be done empirically, however it would suppress all the advantages of the method.

Distance-based method

Maybe the simplest combination method consists in using directly the distances produced by the SVM classifiers to make the final call. This method is typically used along with the one-versus-rest scheme. Once the K classifiers have been trained and evaluated for a given input sample $\mathbf{x}$, the sample is classified as belonging to the class with the largest positive distance as described in [13]. The underlying idea is that samples closer to the decision boundary are more likely to be outliers. However, distances produced by SVMs are uncalibrated values so their direct usage for classification, may not produce satisfactory results. Other, more interesting, strategies will be discussed next.

Voting scheme

This method was initially proposed in [14] and applied to SVMs with excellent results in [15]. The method consists of training pairwise classifiers as described in 2.3.1 and evaluating all of them so each class receives a vote for each positive evaluation. Finally the class with more votes is selected to label the input sample. The main advantage of this scheme if its simplicity, however it considers the output of a SVM as a binary value $-1, 1$, and does not make use of the information that can be potentially extracted from the distance values (i.e. large distances to the separating hyperplane are typically indicators of more reliable predictions).

DDAGs

A Decision Directed Acyclic Graph (DDAG) [16] is a learning architecture that is used to combine many binary classifiers into a multiclass classifier. The DDAG contains $K(K-1)/2$ internal nodes, one for each pairwise classifier trained, and K leaves, where K is the total number of classes. The nodes are arranged so there is a single root node in the top and each non-leaf node has two descendants. To classify an input sample $\mathbf{x}$ belonging to the feature space X, the binary function (SVM pairwise classifier trained as described in 2.3.1) in the root node is evaluated, depending on the result $\{+1, -1\}$ the input is evaluated using the binary function associated to the left or right descendant node, thus excluding one class on each transition. Once a leaf node is reached, the input is classified as belonging to the class associated to the leaf. In Figure 2.3.2 a Decision DAG

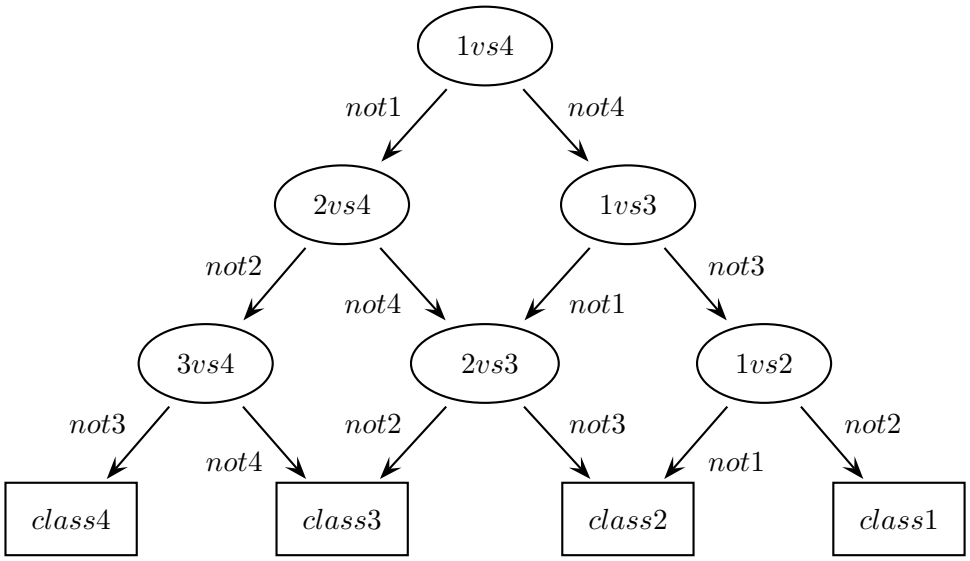

Figure 2.1: *Decision DAG for four-classes classification.*

for four-classes classification is depicted. The internal nodes and the root are represented by ovals tagged with the pairwise classifier attached to them while the leaves of the graph are the set of classes and are represented by boxes.

A Decision DAG is a natural generalization of a class Decision Tree, however since a leaf is reachable by more than one path in the graph, it is not a tree but a DAG. While the binary functions at the nodes of the DAG are the equivalent as that of Decision Trees, the graph representation presents both computational and learning-theoretical advantages [16]. DDAGs guarantee that every time a pairwise classifier is evaluated a class k_i is excluded so only $K-1$ evaluations need to be carried out to label an input sample. For this reason, this method is considerably faster than the voting scheme, where all the $K(K-1)/2$ classifiers need to be evaluated to predict the class of an unlabeled input sample.

Probabilistic approaches

Both the voting scheme and DAGS have proven good performance for multiclass classification, however, while these methods only predict a class label, in many scenarios like speech recognition, a probabilistic estimate for each class k_i given an input sample $\mathbf{x}$ is desired. Once a class-posterior probability $p(k_i|\mathbf{x})$ is computed for each class as described in section 2.2, the classification method

is as follows: given an input $\mathbf{x} \in \mathbf{X}$ where $\mathbf{X}$ is the feature space, it is classified as belonging to the class k which receives the highest probability value. This is expressed in 2.21.

$$k = \underset{k_i, i \in [1,\ldots,K]}{\operatorname{argmax}} \; p(k_i|\mathbf{x}) \tag{2.21}$$

In this section several methods for estimating the individual class-posteriors $p(k_i|\mathbf{x})$ are reviewed.

First, consider the case of having a set of SVMs trained following the one-versus-rest strategy. In this case, class-posterior probabilities $p(k_i|\mathbf{x})$ can be obtained in a straightforward fashion by mapping the distance obtained from each of the K classifiers to a probabilistic value using expression 2.20. However, note that $\sum_i p(k_i|\mathbf{x}) = 1$ is not guaranteed, and in fact it is very unlikely, so a last step of probabilities normalization is needed once all the class posteriors are computed.

Now, the case of obtaining class-posteriors from SVMs trained following the one-versus-one strategy will be considered. In this case, a very simple method consists of applying the voting scheme described in section 2.3.2 and for each class k_i divide the total number of votes received by the total number of pairwise classifiers $K - 1$ trained for that class. This is expressed in 2.22.

$$p(k_i|\mathbf{x}) = \frac{\#votes_{k_i}}{(K-1)} \tag{2.22}$$

Note that, again, these probabilities are unnormalized.

A more convenient method consists of mapping the distance produced by each pairwise classifier $SVM_{(k_i,k_j)}$ using expression 2.20 to produce a "pairwise probabilistic" value $p(k_i|k_j \; or \; k_i, \mathbf{x})$. Several mechanisms have been proposed in the literature [17][18] for combining these pairwise probabilities into a class-posterior probability $p(k_i|\mathbf{x})$. In [19], a comparison of these methods, in addition to the probabilistic version of the voting scheme previously mentioned, is carried out. Results show that, while the voting scheme usually produces unsatisfactory results, the other three methods are very stable having the one proposed in [17] the simplest implementation. This method[5]

is expressed in 2.23.

$$p(k_i|\mathbf{x}) = \left[\sum_{j=1, j\neq i}^{K} \frac{1}{p(k_i|k_j \; or \; k_i, \mathbf{x})} - (K-2) \right]^{-1} \tag{2.23}$$

Note that class-posteriors obtained from 2.23 are unnormalized so a normalization step is needed to make the summation of probabilities across all the classes equal to one. However, as it will be shown in chapter 4, these unnormalized probabilities can be directly used as scores for comparing competing sequences of feature frames in the case of a speech recognition system.

An interesting characteristic of this one-vs-one scheme is that most of the pairwise classifiers that take part in the calculation of the class-posterior $p(k_i|\mathbf{x})$ of an input sample $\mathbf{x}$, have not been trained with samples of the actual class of $\mathbf{x}$. Note that this is not the case of, for example, the one-vs-rest scheme, in which all the models are trained using training samples from all the classes. Particularly, according to equation 2.23, $K-1$ pairwise classifiers need to be evaluated to compute each of the K class-posteriors $p(k_i|\mathbf{x})$. Considering that $p(k_i|k_j \; or \; k_i, \mathbf{x})$ and $p(k_j|k_i \; or \; k_j, \mathbf{x})$ are obtained from the same classifier, $K(K-1)/2$ classifiers need to be evaluated, i.e. the whole set of classifiers trained. However, out of those $K(K-1)/2$ classifiers, only $K-1$ were trained using the actual class of $\mathbf{x}$ so the decision function of the remaining evaluated classifiers is not suitable for separating the sample $\mathbf{x}$ from other samples. This, ultimately means that for a given input sample $\mathbf{x}$, the posterior probabilities $p(k_i|\mathbf{x})$ of most of the classes are calculated using a big number of pairwise-classifiers that produce somehow "unexpected" probabilistic values. Nevertheless, the success of this approach resides in the fact that the class-posterior of the actual class of an input sample $\mathbf{x}$ is indeed calculated using only pairwise classifiers trained using samples of the actual class.

Finally, note that a probabilistic scheme based on DDAGS is also possible. Although under that scheme only the probability of the winning class can be fully computed (using for example expression 2.23). The reason is that the winning class is the only class for which all the pairwise posteriors $p(k_j|k_i \; or \; k_j, \mathbf{x})$ are estimated once the tree has been traversed.

[5]Due to its satisfactory performance and simplicity, it will be the method of choice in this thesis for multiclass probability estimation for continuous speech recognition using SVMs (see chapter 4).

2.3.3 Comparison between the different strategies

While both the one-versus-one and the one-versus-rest strategies are probably the most widely used for multiclass classification, there are substantial differences between them that make either of them more suitable for a particular task. For this reason, a comparison between these strategies will be carried out next. Additionally, some comments on the use of the part-versus-part strategy will be added. The comparison will be performed in terms of classification accuracy and training and testing time.

Classification accuracy

Although both the one-versus-rest and one-versus-one strategies result in very good classification accuracy, the later has been reported to have slightly superior classification performance in a number of tasks [9] [20] [21]. Note that, for standard classification tasks (i.e. class posteriors are not needed), both the voting scheme and DDAGs produce good accuracy results and are the most used. In the case of speech processing, one-versus-one classifiers have shown superior performance to that of one-versus-rest classifiers for phonetic classification [22]. In chapter 4 a comparison of different strategies for multiclass classification will be carried out for the case of MFCC feature vectors classification.

In case that probability estimates are required, if pairwise classifiers are the selected decomposition scheme, the method expressed in equation 2.23 has shown very good results compared to alternative, more complex, probabilistic methods [23]. An additional comparison of different probabilistic methods for solving multiclass problems can be found in [24].

Finally, the part-versus-part approach [12] has shown comparable classification accuracy to the one-versus-one approach when the number of resulting clusters is reduced.

Training time

There are two main factors that strongly influence the time required to train the SVM classifiers involved in a multiclass classification task:

- Number of samples used to train each classifier: it is well known that a strong drawback of SVMs when used along with non-linear kernels is the training time scalability. The training

of a SVM requires the solution of a quadratic problem which time complexity is superlinear with the number of samples. Lots of efforts have been made and described in the literature in order to optimize the computation of the solution to the quadratic problem, however this is still a big drawback of SVMs[6].

- Number of classifiers that need to be trained: it is obvious that, assuming that the classifiers are trained on datasets of similar size (and complexity), the training time grows linearly with the number of classifiers that need to be trained.

Table 2.1 summarizes these properties for both strategies discussed, where N is the number of training samples and K is the number of classes.

Training strategy	Number of classifiers	Number of training samples
one-versus-rest	K	N
one-versus-one	$K(K-1)/2$	$2N/K$ [7]

Table 2.1: *Comparison between the one-versus-rest and the one-versus-one strategies in terms of number of classifiers and training samples per classifier.*

In [9] a comparison in terms of training time was reported for different datasets, showing that the one-versus-one clearly outperforms the one-versus-rest approach. This difference is highlighted in datasets for which the number of classes is considerable, which causes a bigger difference between the size of the training sets obtained following either approach. Similar results have been reported in other works, thus indicating that the superlinear complexity of solving the quadratic problem on larger datasets dominates the training time. For this reason, the one-versus-one strategy is more appealing when dealing with large datasets for which the training time of the one-versus-rest classifiers can be either huge or just intractable.

Training time is one of the most appealing features of the part-versus-part decomposition method. As reported in [12], and assuming that a multiprocessor machine is available for carrying out the training (i.e. it is possible to train several classifier in parallel), the training time can be considerably reduced as the subsets of samples decrease in size. Nevertheless, as reported in [12]

[6]This will be discussed in section 7.1 for the case of speech-processing and feature vectors classification.

[7]Note that, since the distribution of the data across different classes depends on the task, this is an expected value.

this can only be achieved at the expense of a reduction in the classification accuracy. This point and the fact that it is not clear how to determine the stopping criterion for the splitting, may be the reasons that have prevented this method from becoming more popular.

Testing time

When it comes to measuring the testing time of different approaches the first consideration is that, in general, the testing time is dominated by the number of kernel evaluations $K(\mathbf{x}_1, \mathbf{x}_2)$. In particular, the testing time can be considered as linear with the number of unique support vectors that need to be evaluated to classify a given input sample. Given that a support vector is always a training sample, classifiers trained with samples from the same class can potentially share a big number of support vectors. This circumstance can be exploited in order to reuse kernel predictions during classification as well as other speed-ups as will be shown in section 7.3.1. It is for that reason that the number of unique support vectors, and not the total number of support vectors obtained from the training of the different classifiers, is what really determines the testing time. However, in some scenarios (and that is the case of cepstral features classification as will be described in section 4), in order to obtain the best classification accuracy, pairwise classifiers need to be trained using different kernel parameters what prevents a direct reuse of kernel evaluations.

For example, when the number of classes K is big, the one-versus-one strategy produces a considerable higher number of support vectors compared to the one-versus-rest strategy, however many of those support vectors are identical so only participate in a single kernel evaluation that is reused across classifiers' evaluations. In this scenario, and always under the assumption that a big number of classes are involved, the number of Lagrange multipliers (which is linear with the total number of support vectors disregarding repetitions across classifiers) also plays an important role in the testing time.

In the case of one-versus-rest classifiers, disregarding the technique employed (i.e. using the distance directly or mapping it to a probability value) all the classifiers need to be evaluated to predict an input sample. However, in the case of one-versus-one, the utilization of DDAGs only implies the evaluation of $K-1$ classifiers to label an input sample, thus resulting in a big reduction of testing time. As reported in [9], this reduction becomes more significant for larger values of K.

2.3.4 Summary and conclusions

The application of SVMs to multiclass classification tasks has received the focus of research effort in recent years. As a consequence of that a good number of techniques have been developed and evaluated in a wide range of scenarios. In this section some of the most important ones have been analyzed from different perspectives: classification-accuracy, trainability and evaluation-time. Decomposition techniques have shown very good performance in a number of tasks and are by far the most utilized. While the one-vs-one technique presents trainability and classification accuracy advantages, it produces a number of classifiers that is quadratic with the number of classes. In the other hand, the one-vs-rest strategy exhibits slightly worse classification accuracy and serious trainability issues when dealing with large datasets. However, it presents clear advantages in terms of evaluation time. The suitability of both techniques is mostly task-dependent and will be discussed in the next chapters.

Chapter 3

SVMs for Speech Processing

Support Vector Machines (SVMs) [1] are a well-established machine learning technique that, due to their remarkable generalization performance, have attracted much attention and gained extensive application in many fields including speech processing. Unlike other traditional techniques like Artificial Neural Networks (ANNs), SVMs perform both an empirical and a structural risk minimization over the training set, resulting in better generalization.

The idea of modeling the acoustics of speech using discriminative classifiers already showed very appealing results in the case of ANNs [25] [26]. Nevertheless, ANNs-based techniques, have shown some limitations when applied to speech processing [27] [28] that have impeded their widespread adoption and have made them fail to replace conventional GMM/HMM approaches. Among their limitations are the following:

- ANNs tend to overfit the training data, which compromises their generalization ability. This happens when the capacity of the network significantly exceeds the needed free parameters.
- Slow convergence during the learning. This is typically addressed by carefully selecting a big enough gradient and an adequate network topology.
- Search space may present multiple local minima.
- The selection of an optimal network topology (number of layers and units) may not be easy and may vary with the application.

In the other hand, maximum margin classifiers, like SVMs have shown a very good generalization performance[8] and only a few training parameters need to be selected (i.e. the error penalty C and the kernel parameters). Additionally the training of SVMs guarantees to find the optimal solution (see chapter 2). What also makes SVMs more appealing than ANNs is that the capacity of the model is automatically determined during the training process and it depends only on one parameter (the regularization parameter C) that controls the trade off between misclassification on the training set and the complexity of the decision function. This parameter can be easily estimated from the training material doing cross validation. In the other hand, while finding a convenient neural network topology is not straightforward and typically is task-dependent, selecting a suitable kernel in the case of SVMs may be seen as an analogous problem when no prior information of the input data is available. Nevertheless, polynomial and gaussian kernels have shown very good performance in most scenarios.

However, the connection between ANNs and SVMs is undeniable and, in fact, a nonlinear SVM using a sigmoid kernel (see section 2.1.2) is completely equivalent to a two-layer neural network in which the optimal number of units in both layers is automatically determined during the training process. Nevertheless, SVMs still present some drawbacks, among which, the training time scalability and the complexity of the decision function (that is expressed as a function of a subset of the training samples) are maybe the most critical.

First applications of SVMs to speech processing were focused on simple classification tasks, like phonetic classification [22] in which SVMs already showed very promising results. Afterward, SVMs have also been successfully applied to other classification tasks like rejection [29] [30] or speaker verification [31]. In recent years, several Automatic Speech Recognition (ASR) systems have been proposed, in which SVMs are used as statistical estimators rather than just for standard pattern classification. While some of these systems are just used for hypothesis rescoring on top of a conventional GMM/HMM decoder [32] [33], other systems [34][35][36], make use of SVMs during the decoding process to replace Gaussian Mixture Models (GMMs) in the calculation of emission probabilities at each of the states of a given word-level or phone-level HMM. For example, in [35], a

[8] In recent years, motivated by the excellent generalization performance of large margin classifiers, very successful discriminative training techniques to estimate the parameters of HMMs have arisen. This will be commented later on in this chapter.

hybrid system based on word-level HMMs is used for digit recognition under noisy speech showing a superior performance compared to a conventional GMM/HMM paradigm.

This chapter is dedicated to outline the state-of-the-art on the application of SVMs to speech processing as well as discussing the most promising techniques and their target scenarios.

3.1 Modeling the temporal structure of speech

One of the main difficulties that speech processing techniques face, and so is the case of SVMs, is how to deal with the variable length of the speech units (phones, syllables, words, etc) under consideration. Typical techniques used for speech parameterization (like MFCC, LPC or PLP) use a fixed length window (of about 25ms) to extract feature vectors of fixed dimension. Consecutive feature vectors are computed by shifting the window (typically at intervals of 10ms) until the end of the parameterized utterance is reached. Given this parameterization scheme, it is clear that speech units of different length result in variable length sequences of feature vectors. It is well known that this problem, also present in other temporal pattern recognition tasks like handwriting or gesture recognition, can be addressed using HMMs.

In the case of SVMs, which assume that the input samples are feature vectors of fixed dimension, several techniques have been proposed to integrate them into the classification of variable length sequences of feature vectors. This section is dedicated to outline some of the most relevant techniques proposed to date and establish a comparison between them.

3.1.1 Normalization methods

These methods consist in normalizing the variable length sequence of feature vectors into a fixed length feature vector that can be used as input for the SVMs. An example of normalization method is the averaging method proposed in [32], that lies in the assumption made by the HMM modeling that speech segments (phones in [32]) are composed of a fixed number of sections. Under this assumption the idea consists in using the time-alignment information produced by a conventional HMM/GMM Viterbi aligner to extract phone-level feature vectors. These vectors are obtained by averaging the feature vectors (typically MFFC vectors) corresponding to time frames aligned

to each of the HMM states, and concatenating these average vectors to create a composite vector representing the phone. Note that, the resulting phone-level feature vectors have a dimensionality n times larger than the original feature-vectors, where n is the number of HMM-states. While this averaging approach results in a very compact representation of the training data, allowing for a fast training of the SVM models, it can't be decoupled from the GMM/HMM system that produces the state alignments. Additionally, two systems need to be trained on the same data, a GMM/HMM system and the set of SVMs. SVM-based systems like this, which work in conjunction with a conventional GMM/HMM system, will be referred in the following as hybrid systems. A clear drawback of hybrid systems is that they rely on the alignment made by a conventional GMM/HMM system so errors resulting from a bad alignment can hardly be recovered by the SVM classifiers.

3.1.2 Fisher-kernels

A very popular technique for dealing with the variable length of speech sequences is the Fisher Kernel [37]. This technique combines the advantages of generative statistical models (like the Hidden Markov Models) and those of discriminative methods (like Support Vector Machines). The Fisher Kernel approach has been successfully applied to speech signals in several tasks like audio classification [38] or speaker verification [39] [31]. In the case of speech, the idea consists of using GMMs to calculate emission probabilities as an intermediate step to generate fixed-length feature vectors, which can be used as input for a SVM classifier. Given an utterance composed by the sequence of feature vectors $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ and assuming that each vector in the sequence is independent and equally distributed, the Likelihood of the utterance can be defined as $P(X|\theta) = \prod_{i=1}^{l} p(\mathbf{x}_i|\theta)$, where each $p(\mathbf{x}_i|\theta)$ is obtained from the corresponding mixture of gaussians. The procedure of mapping an utterance X (that can be a phone, word, sequence of words, etc) to a fixed-length vector is carried out using the Fisher score, as expressed in 3.1.

$$U_X = \nabla_\theta \log P(X|\theta) \tag{3.1}$$

Each of the parameters of the fixed length vector U_X is the derivative of the log-likelihood of the utterance X with respect to a given parameter of the generative model (mixture of gaussians).

Once a fixed length feature vector has been obtained for each utterance using the Fisher score, these vectors can be used to train a SVM classifier.

This procedure presents similar drawbacks to those of normalization methods, i.e. necessitates the training of generative models in addition to the SVMs. Nevertheless it has shown very good results on several tasks.

3.1.3 SVM/HMM approaches

A very natural technique to deal with variable-length feature vectors consists of using SVMs classifiers as probabilistic estimators to compute class-conditional probabilities (also known as emission probabilities) while preserving the utilization of HMMs to model the time-varying structure of speech (i.e. SVMs are used as statistical estimators replacing the widely adopted mixture of Gaussians). A complete procedure for the calculation of emission probabilities using SVMs will be described in the next chapter.

This notion of using discriminative classifiers to estimate emission probabilities shares some of the motivations of using discriminative training (and specifically large-margin discriminative training (LMDT)) instead of Maximum Likelihood Estimation (MLE) for training the parameters of HMMs. Some of these motivations are listed below:

1. GMMs trained following the Maximum Likelihood Estimation criterion do not provide optimal speech classification because the real distribution of speech data is unknown.

2. MLE does not directly minimize word or phoneme recognition error rates and often does not provide the decision boundaries in terms of minimizing the error rates. A discriminative training criterion is, thus, more closely related to the recognition error rate.

Different discriminative training criteria have been adopted over the last decade to replace the MLE criterion in the training of HMMs parameters for improved accuracy. While some of these methods: Maximum Mutual Information [91], Minimum Classification Error [92] or Minimum Phone Error [93] are based on the minimization of the classification error on the training set, recent techniques based on large-margin discriminative training (LMDT) (see [94] for a complete review of these techniques) allow better generalization on the test set. Given that SVMs are a very

well known example of the excellent generalization performance resulting from large margin based training, they seem to be very good candidates to work as probabilistic estimators for emission probabilities.

3.1.4 Comparison between the different methods

Maybe the main advantage of the SVM/HMM approach is that it is especially suitable for continuous speech recognition, where the starting and ending times of the units under recognition (words, phones, etc) are unknown in advance. Since most state-of-the-art ASR systems are based on HMMs, this point needs no further justification. In the other hand, note that normalization methods as the one described in section 3.1.1 or the Fisher Kernel, can only be applied on a variable-length sequence of feature vectors which starting and ending time frames are previously known. In such a situation, building an ASR system using either of those techniques would require considering all the hypothetical starting and ending times for a given sequence of feature-vectors without reusing computations, which seems to be impractical. Summarizing, HMMs assume independence between adjacent feature vectors, which allows the application of dynamic programming techniques and thus reusing computations at the frame level, which is of major importance in continuous speech recognition.

However, when it comes to classification tasks (like phonetic/audio classification or speaker verification) the SVM/HMM approach is less appealing than Normalization approaches or the Fisher Kernel. The reason is that a SVM/HMM system, as it is the case of a GMM/HMM system, makes the assumption of conditional independence between adjacent feature vectors. In the other hand, normalizations schemes or the Fisher kernel take advantage of the correlation of adjacent feature frames.

3.2 An example on the utilization of SVMs for speech processing

In this section, in order to highlight the excellent performance of SVMs in speech processing tasks, a novel application of SVMs to speech classification will be described. The application is based on a novel method for automatic pronunciation error detection of childrens speech. The next subsections

are dedicated to introduce the method proposed, describe the experimental setup, show the results and finally, draw conclusions.

3.2.1 Introduction

Typical speech recognition systems tend to average across pronunciation variances in order to minimize the word error rate (WER). However, in some educational situations, reading and language learning in particular, the pronunciation of a word may be of the utmost importance. For example, if children are asked to say the opposite of the word "down", their semantically different responses should be easy to judge with current speech recognizers (e.g. "up," "low," "there"). But, if children are asked to read the word "down," their phonologically similar responses require a different kind of system, a pronunciation error detection system that can discriminate the correct response from phonologically close errors (e.g. "down," "drown," "dawn"). Work in our Center (Center for Computational Language and EducAtion Research, (CLEAR)) in assessing and remedying reading difficulties will benefit enormously from such a system [95]. A computer system can easily and reliably score childrens word reading independently in forced-choice exercises where the program pronounces a word, and the child chooses the correct item among carefully chosen distractors. However, reading recognition and reading production are different processes, and an assessment system limited only to recognition would be an impoverished one. In the following it will be described a pronunciation error detection technique in the context of a word reading task, whose goal is to develop an Independent Comprehensive Adaptive Reading Evaluation called ICARE [96].

Previous work in pronunciation scoring [97][98] has shown that scores based on phone-level posteriors outperform alternative scores that make use of normalized phone duration or Hidden Markov Models (HMMs) log-likelihood. In [97] a measure for the quality of the pronunciation of a phone, called GOP (Goodness Of Pronunciation) was introduced. As expressed in 3.2, this measure consists of normalizing the logarithm of the phone-level posteriors by the phone duration. NF is the phone duration in terms of the number of frames aligned to the phone.

$$GOP(ph|\mathbf{X}) = \frac{|logp(ph|\mathbf{X})|}{NF(ph)} \tag{3.2}$$

According to Bayes rule 3.3 phone posteriors $p(ph|\mathbf{X})$ can be calculated from the acoustic

scores $p(\mathbf{X}|ph)$, the phone prior probabilities $p(ph)$ and the probability of the sequence of observations $p(\mathbf{X})$.

$$p(ph|\mathbf{X}) = \frac{p(\mathbf{X}|ph)p(ph)}{p(\mathbf{X})} \tag{3.3}$$

Unfortunately $p(\mathbf{X})$ is unknown so in the case of pronunciation scoring [97][98] phone posteriors are usually estimated normalizing the acoustic scores as expressed in 3.4. This procedure consists of dividing the acoustic score of the target phone by the acoustic score of the competing phones 3.3. P is the set of phonetic symbols.

$$p(ph_i|\mathbf{X}) = \frac{p(\mathbf{X}|ph_i)p(ph_i)}{\sum_{ph_j \in P} p(\mathbf{X}|ph_j)p(ph_j)} \tag{3.4}$$

Given its proven good performance this measure has been selected as the baseline feature for the system.

The pronunciation scoring system proposed here uses information contained in phone graphs to calculate three different phone-level features, which are ultimately combined to produce word-level pronunciation scores. The features are the following:

- Phone posteriors calculated over phone graphs using the forward-backward algorithm as described in [44]. This is an alternative method to 3.4 for calculating phone posteriors using HMM/GMM acoustic scores.
- Phone posteriors calculated using SVMs trained with segmental features (See section 3.1.1).
- Phone-level scores obtained from frame-level posteriors. Pairwise SVM models trained with cepstral features are used at the frame-level to calculate the posteriors of each of the competing phone classes in the graph.

These phone-level features, in addition to the GOP feature used as baseline, are ultimately used by the scoring module to produce word-level pronunciation scores and make the final decision of whether to accept or reject the word pronunciation. The system is divided into three modules: the phone graph generation module, the features extraction module and the scoring and classification

module. These modules are described in the following sections.

3.2.2 Generation of the phone graph

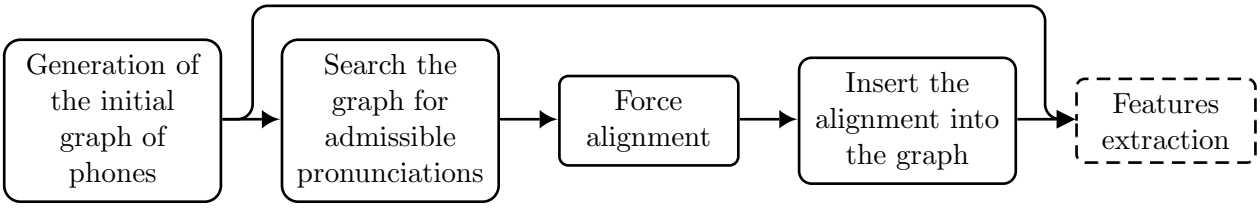

Figure 3.1: *Block-diagram of the graph generation process.*

For each audio file containing a word realization, a phone graph is generated containing at least one admissible pronunciation (phonetic transcription) of the word to score in addition to other sequences of phones that align with a high likelihood to the speech segment. Graphs containing high likely competing sequences of phones allow the extraction of high quality features as will be described in the next section. The block-diagram corresponding to the phone graph generation process is depicted in figure 3.1. Initially, Sonic, the speech recognition system [41] is used to generate a phone graph. Given that many phone sequences are not possible in the English language, the search is slightly constrained using a discounted phone-based trigram language model. The language model is trained using text corpora extracted from English text books and the admissible pronunciations of the 138 target words included in the vocabulary. The acoustic models are trained on a corpus of read children's speech [42]. Given that the intention of the system is to score children's pronunciation quality with respect to children's native pronunciation quality, only native speakers are used to train the acoustic models. The graph of phones produced is expected to contain admissible pronunciations of the word to score. However, disregarding the quality of the pronunciation, the phone graph does not always include a phonetic transcription of the word to score. Using a set of phonetic transcriptions selected by linguists to capture admissible pronunciations of the word to score, a multiple pronunciation (MP) phone graph is built. By doing a text alignment between the MP graph and the one generated by Sonic it is possible to check if the later contains at least one admissible pronunciation of the word to score. The algorithm used to perform the alignment is very similar to the one proposed in [43], however, since no edit

errors are allowed, the algorithm performs very fast. In case no pronunciation of the target word is found, a forced alignment between the audio segment and the graph of multiple pronunciations is carried out using the align tool of Sonic. State alignment information and phone-level acoustic scores resulting from the alignment are used to insert the alignment in the phone-graph.

Finally, note that this procedure allows the insertion of typical mispronunciations of a word used as distractors that can be obtained using language specific rules or information extracted from the training data.

3.2.3 Features extraction

Several features have been selected for scoring the word pronunciations. In this section it will be discussed the process of extracting those features.

Phone-graph based posterior probabilities

Following the process described in [44] a posterior probability has been generated for each of the phones contained in the graph. The posterior probability of a phone $p([ph; s, e]|\mathbf{X})$ can be calculated as expressed in 3.5 by summing up the posterior probabilities of all paths in the graph of length M which contain the hypothesis $[ph; s, e]$. Where $[ph; s, e]$ is the phone starting at time s and ending at time e, and $\mathbf{X} = \{x_1, ..., x_T\}$ is the acoustic observation sequence against which it is aligned.

$$p([ph; s, e]|x_1^T) = \sum_{\substack{[ph;s,e]_1^M:\exists n\in\{1,...,M\}:\\ [ph_n;s_n,e_n]=[ph;s,e]}} \frac{\prod_{m=1}^{M} p(x_{s_m}^{e_m}|ph_m)^{\alpha} p(ph_m|ph^{m-1})^{\beta}}{p(x_1^T)} \tag{3.5}$$

In the following $p([ph; s, e]|\mathbf{X})$ will be used as a phone-level feature referenced as C_m. While the use of a phone-based language model helps in the lattice generation process given that many phone sequences are not possible in the English language, the language model probability has not been used for the calculation of the posteriors, so the scaling factor β in 3.5 receives a value of 0.

SVM based phone posteriors

As introduced at the beginning of this chapter, previous work has shown that SVMs can be used as reliable estimators of the posterior probability of a speech segment [45][46]. SVMs can be used as probabilistic estimators by mapping the margin or distance they produce to a posterior class probability using a sigmoid [47]. As previously mentioned in section 3.1, one of the main difficulties when applying SVMs to speech classification is how to deal with the variable length of the speech units under classification. Fisher Kernels have been proposed to address this problem using GMMs as an intermediate step to generate fixed length feature vectors to be used as input for the SVM. However, in the case of phones, a very straightforward technique has shown very good performance [45]. This technique consists of averaging the feature vectors (typically cepstral features) aligned to each of the states of a phone and then concatenating the average vectors to create a composite vector. The resulting composite vector has a dimensionality three times larger than the original vectors. For simplicity, the second strategy has been selected; it is expressed in 3.6.

$$p_{segments}(ph|x_1^T) = p(ph|composite(x)) \tag{3.6}$$

SVMs are trained following a one-vs-rest approach so one SVM is trained for each of the phonetic classes used.

SVM based frame level posteriors

In this case, SVMs have been used to estimate posteriors at the frame level using only frame-level competing candidates in the graph. For each time frame, the phone classes aligned to that frame in the phone graph are used to calculate the frame posteriors. According to the state alignment information present in the graph, each time frame can be aligned to one out of K different classes (that correspond to each of the HMM states of the phonetic symbols used). Given that for a given time frame only a (typically small) subset K' of the K classes is present in the graph, a one-vs-one strategy has been selected for training the SVMs. One-vs-one SVMs learn the decision boundary between two classes, so a different model needs to be trained for each pair of classes. Thus, the resulting number of SVMs is $K(K-1)/2$. If k_i is one of the K' classes, the posterior probability

of k_i given an observation $\mathbf{x}$ can be computed (as previously introduced in section 2.3.2) using equation 3.7.

$$p(k_i|\mathbf{x}) = \left[\sum_{j=1, j\neq i}^{K} \frac{1}{p(k_i|k_j \text{ or } k_i, \mathbf{x})} - (K' - 2) \right]^{-1} \tag{3.7}$$

Finally, as expressed in 3.8, a confidence measure C_f is calculated for each phone by averaging its frame level posteriors. S represents the number of states of the HMM.

$$C_f(ph|x_1^T) = \sum_{s=1}^{S} \sum_{t=t_1}^{t_s} p(k_{ph,s}|x_t) \tag{3.8}$$

3.2.4 Experimental procedure

Speech Material

The speech material used in the experiments is divided in two parts. The first part, used exclusively for training purposes, consists of about 4 hours of read speech of first graders (171 different speakers) from the CU Read and Summarized Story Corpus [42]. This corpus has been used to train the HMM/GMM acoustic models used by Sonic to generate the phone graphs and to train the phone-level and frame-level SVM classifiers described in sections 3.2.3 and 3.2.3 respectively.

The second part is a corpus of read words annotated for pronunciation scoring. It consists of 2340 pronunciation instances of 138 unique words from 99 poor readers in Kindergarten through 5th grade. Readers ages ranged from 5½ to 11 years old with average 7 years. There were 55 males and 44 females.

The pronunciation instances were transcribed at the phone level with 1816 scored as correct and 524 scored as incorrect by two experts. For the task, the 99 readers had been presented printed words to read, one at a time, with no time limit, by the ICARE program. Words ranged from easy monosyllabic letters and orthographically regular or high frequency words, to more challenging multisyllabic and orthographically irregular ones. Children began with words estimated to be at their reading level from another test [48], and then the program used a skipping algorithm that established a basal of 5 letters or words in a row correct at the low end of the list, presented all words in order in the middle, and continued until 5 of the last 7 words were missed to establish a

ceiling level. Since this corpus may be considered too small to achieve reliable speaker-independent results, we have used 5-fold cross-validation to artificially extend it, averaging the results afterward. Particularly, we have divided the corpora into 5 balanced speaker-disjoint subsets. One different subset is kept for testing in each fold, while the remaining ones are used to train the word-level classifiers described in previous sections.

SVMs training

In this section, the training process of the phone-level and frame-level SVMs described in sections 3.2.3 and 3.2.3 respectively is briefly described. For every speech utterance present in the training set, 39-dimensional feature vectors, consisting of 12 Mel Frequency Cepstral Coefficients and energy plus first and second order derivatives, have been extracted. Using a Viterbi aligner, segmental and frame level features have been extracted and used to train the SVMs. The SVM-library used is LibSVM [49]. A radial basis function (RBF) kernel is used for which the parameters C (error penalty) and γ are estimated on the training set following a "grid-search" process using 5-fold cross validation.

Evaluation Metric

For each word, phone level features are extracted as described in section 3 and a classifier as described in section 4 attaches a correctly pronounced tag if the word-level pronunciation score is above the fixed threshold or an incorrectly pronounced tag otherwise. The metric used to evaluate the performance of the system is the Detection Error Rate (DER), defined as the total number of incorrectly assigned tags divided by the total number of scored words:

$$DER = \frac{\#incorrectly\ assigned\ tags}{\#scored\ words} \tag{3.9}$$

Results

The experiment carried out evaluates the quality of the phone-level features proposed for pronunciation scoring in terms of DER. Four sets of SVMs like the ones described in section 4 are trained to produce word-level pronunciation scores using each one of the three phone-level features proposed

Feature	DER	Relative error reduction
GOP (baseline)	0.2013	
C_m	0.1987	1.29%
C_f	0.1772	11.97%
$P_{segments}$	0.1584	21.31%
C_f and $P_{segments}$ combined	0.1552	22.90%

Table 3.1: *Detection Error Rate for the system using each of the features proposed and the baseline feature.*

in section 3 plus the GOP feature. These SVMs are used to score each one of the 2340 words in the test set following the cross-validation process described in section 4.1.1. Table 1 shows DER results using each of the features separately.

As can be seen in Table 3.1, two of the proposed features clearly outperform the baseline. However the C_m feature yields a very similar performance compared to the baseline. This is not surprising since both of them are based on phone-level posteriors calculated doing a normalization of HMM/GMM acoustic scores over competing phone candidates. On the other hand, the SVM-based features (C_f and $P_{segments}$) clearly outperform the baseline with a 11.97% and 22.90% relative error reduction, respectively. Figure 3.2 shows a Detection Error Tradeoff curve for each of the features. This curve shows the percentage of false acceptations (mispronunciations tagged as correct pronunciations) against the percentage of false rejections (correctly pronounced words tagged as mispronunciations) for different values of the threshold. As can be seen the new proposed features significantly increase the system operating range.

3.2.5 Conclusions

It has been introduced a novel technique for pronunciation error detection based on the combination of several phone-level features extracted from phone-graphs. These features have been combined using SVMs to produce word-level pronunciation scores that can be used to determine whether a word has been correctly pronounced.

Pronunciation scores computed following these procedures have shown superior performance compared to state-of-the-art ones. In particular, the combination of two of the features proposed achieves a relative error reduction of 22.9%. From these results it can be concluded that phone-

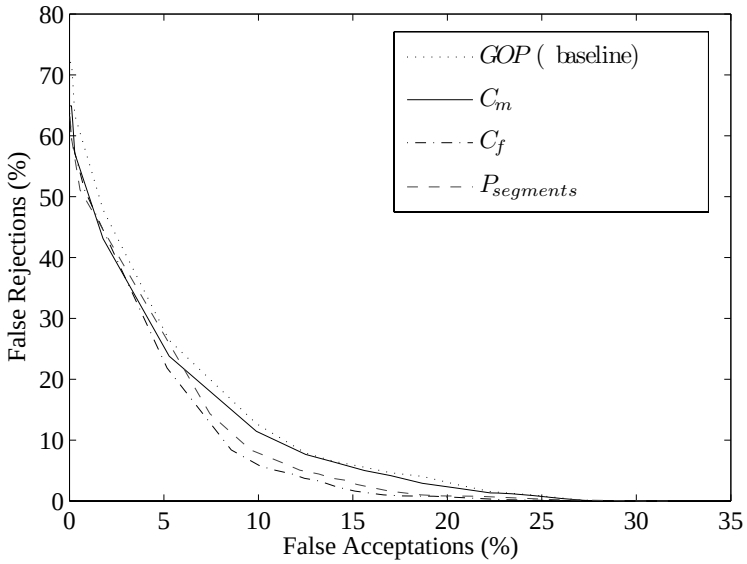

Figure 3.2: *Detection-error-tradeoff curves for each of the features proposed and the baseline.*

graphs are a suitable representation of competing sequences of phones for a given speech utterance, that can be effectively used to extract high quality phone-level features to be used for pronunciation scoring.

From the point of view of SVMs applied to speech processing, the proposed technique shows that SVM-based features can be effectively utilized in a variety of speech classification tasks.

As far as the pronunciation error detection technique (as a whole) is concerned, future work will aim to improve the detection rate and further detect not only whether a childs reading response is correct, but also whether the error is among the most common error patterns. This information can provide diagnostic information about common patterns such as sound reversals, additions, deletions, or substitutions that can help guide instruction by teachers or programs.

Chapter 4

SVMs for Continuous Speech Recognition

4.1 Introduction

Although a growing number of speech processing applications have benefited from the use of SVMs (especially for classification tasks) its effective application to continuous speech recognition in a stand-alone system is still pending. In this chapter, a basic architecture for a SVM/HMM continuous speech recognition system will be described. Additionally, experiments will be carried out in order to evaluate this architecture and a discussion will take place in order to expose its advantages and disadvantages respect to conventional GMM/HMM systems.

Nowadays, the vast majority of state-of-the-art Automatic Speech Recognition (ASR) systems model the acoustics of speech using GMM/HMM acoustic models. The speech signal, can be viewed as a piecewise stationary signal or a short-time stationary signal, thus, it is possible to assume that in a short period of time in the range of 10 milliseconds, speech can be approximated as a stationary process. These small segments of speech can thus be encoded into real-valued feature vectors of fixed dimension using a wide variety of techniques (MFCC, LPC, PLP, etc). Left-to right HMMs (see figure 4.1) are then used to estimate the probability that a sequence of these vectors is produced in the realization of the phonetic or lexical units (phones, syllables, words, etc) used in the recognition. At each of the HMM-states, the probability of observing a feature vector

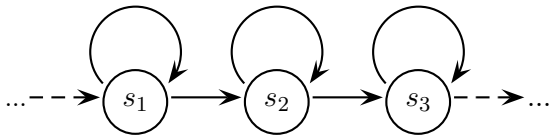

Figure 4.1: *Three-state left-to-right Hidden Markov Model.*

(emission probability) is typically calculated using a mixture of diagonal covariance Gaussians. The probabilities of transitioning between HMM-states (transition probabilities) along with the emission probabilities can be estimated using the Expectation Maximization Algorithm (EM) [50] [51]. During decoding, the estimation of the probability of a sequence of phonetic or lexical units is carried out concatenating the corresponding HMMs into a composite HMM.

As it was previously introduced in chapter 3, a natural mechanism for the application of SVMs to continuous speech recognition consists in using them as statistical estimators of emission probabilities instead of GMMs, which results in a SVM/HMM system. Based on that idea, a hybrid SVM/HMM continuous speech recognition system was proposed in [36] in which GMMs are replaced by SVMs in the computation of the emission probabilities of each of the states of phone-level HMMs. While such system yields better recognition accuracy than a comparable[9] GMM/HMM speech recognizer, it still presents several limitations:

1. It relies on the transition probabilities between HMM-states obtained from acoustic models trained using a conventional GMM/HMM system. This connection with a GMM/HMM system (that needs to be trained in advance) makes this system fall into the category of hybrid systems.

2. It does not make use of phonological similarity information when training the classifiers.

3. The one-versus-one strategy that was selected for training the SVM classifiers, while outperforming other strategies in terms of word accuracy [52], it requires the evaluation of several thousand pairwise classifiers at any given time frame during decoding.

4. Context dependency was not modeled, furthermore, no such attempt has ever made in the literature in the case of SVMs and continuous speech decoding, so it is not clear how to address this issue.

[9]The system proposed in [36] does not model context dependency so it was compared to a monophones conventional system.

In this chapter a stand-alone SVM/HMM system that overcomes some of the limitations[10] of the system proposed in [36] will be described and evaluated. Some of the main characteristics of the proposed system are the following:

- It does work as a stand-alone system. In fact, its only connection with a GMM/HMM system is the state-alignment required in the segmentation of the training data used to train the SVM classifiers. However this connection is not that because hand-labeled speech corpus like TIMIT can potentially be used to generate a set of bootstrap models.
- It performs speech recognition and not hypothesis rescoring (in contrast with other systems like [32]).
- It is a continuous speech recognition system (in contrast with isolated recognition approaches like [34] or [35]).
- It does not make use of transition probabilities extracted from acoustic models trained using a conventional GMM/HMM system (in contrast with the system proposed in [36]). For this reason, it can not be considered a hybrid approach but an independent system.

4.2 System's Description

The main difference between the SVM/HMM ASR system proposed in this chapter and a conventional GMM/HMM one lies in how the acoustic modeling is carried out. This section is devoted to describe the acoustic modeling in terms of how the emission probabilities $p(\mathbf{x}|k_i)$ of each of the HMM-states k_i are calculated, how transition probabilities are modeled and what are the topologies selected for the HMMs.

4.2.1 Emission probabilities computation

Calculating emission probabilities for HMM-states using SVMs comprises basically two steps. First, given that SVMs are binary classifiers, a multiclass-classification strategy must be adopted to separate samples belonging to different HMM-states (which number depends on the size of the phonetic

[10] Other limitations of that system, like the use of phonetic similarity, the training/decoding time scalability and the context modeling, will be addressed in the following chapters.

symbols set and is obviously higher than two). However, it is not enough with attaching a class (HMM-state) label to a given sample $\mathbf{x}$ as in standard classification, but a class-posterior probability value $p(k_i|\mathbf{x})$ needs to be computed for each of the classes k_i comprised in the classification process (i.e. for each of the HMM-states that are active in the Viterbi search at time t). Once these class-posteriors are computed, they need to be transformed into class-conditional probabilities (also known as emission probabilities) $p(\mathbf{x}|k_i)$, which constitutes the second step.

This section is dedicated to the description of both steps.

Multiclass classification using SVMs

SVM classifiers are binary classifiers so for K-class ($K > 2$) separation tasks, like separating samples belonging to different HMM-states, several SVM classifiers need to be trained. In particular, having a phonetic symbol set of size P and S states for each of their corresponding HMMs, $K = PS$ classes need to be discriminated. Note that only a subset K' of the K HMM-states are active at any given time frame in the Viterbi search, however, for simplicity, this consideration will be ignored in this explanation. As described in section 2.3 there exist several methods for multiclass-classification. Among all of them, the one-vs-one method based on training a pairwise classifier for each pair of classes k_i and k_j (i.e. a total number of $K(K-1)$ classifiers need to be trained) has been selected, the reasons are the following:

1. Pairwise classifiers have shown superior or at least comparable classification performance in comparison with alternative strategies (see section 2.3) in a number of pattern recognition tasks.

2. The pairwise scheme produces classifiers that, unlike those produced by the one-vs-rest scheme, are trained on only a fraction of the whole training dataset, which can be handled considerably more efficiently. Recall from section 2 that SVMs require a training time superlinear with the number of samples. However, as it will be explored in section 7.1, a very recent SVMs' training technique allows a very efficient training in very large datasets what might make one-vs-rest classifiers more attractive in this respect.

3. Pairwise classifiers allow a flexible decoding process by facilitating the removal of some of the

classifiers from the emission probability calculation process. As it will be detailed in the following chapters, this fact can be exploited for doing state-tying and for dynamically selecting the pairwise classifiers that contribute to the emission probabilities calculation process.

4. One-vs-one classifiers have already shown very good performance in hybrid SVM/HMM ASR systems [35].

SVMs as probabilistic estimators

Under the pairwise framework, given an input sample $\mathbf{x}$, each of the pairwise classifiers trained is used to obtain a pairwise class-posterior $p(k_i|k_j \; or \; k_i, \mathbf{x})$. The procedure of mapping the output of the SVMs to these probabilistic values was described in section 2.2. Those pairwise probabilistic values can then be used to calculate the posterior probability $p(k_i|\mathbf{x})$ of each of the K classes (HMM-states). This will be carried out, as described in section 2.3.2, using the following expression:

$$p(k_i|\mathbf{x}) = \left[\sum_{j=1, j\neq i}^{K} \frac{1}{p(k_i|k_j \; or \; k_i, \mathbf{x})} - (K-2) \right]^{-1} \tag{4.1}$$

according which only the pairwise classifiers trained for a class k_i contribute to the computation of its class-posterior.

Moving from class-posteriors to class-conditioned probabilities

In a conventional GMM/HMM ASR system, a mixture of gaussians allow for a direct calculation of the emission probabilities, this can be summed up in expression 4.2 for the case of continuous mixture density HMMs, where b_j denotes a single Gaussian density function and c_j is the weight for the jth mixture component subject to the constraint 4.3. All of those parameters along with the transition probabilities between HMM-states can be efficiently calculated using the EM algorithm [50] [51].

$$p(\mathbf{x}|k_i) = \sum_{j=1}^{M} c_j b_j(\mathbf{x}) \tag{4.2}$$

$$\sum_{i=j}^{M} c_j = 1 \tag{4.3}$$

In the case of discriminative classifiers, like ANNs or SVMs, the situation is a little bit more complex since they estimate class-posteriors $p(k_i|\mathbf{x})$ that, although very useful for classification tasks, need to be transformed into class-conditional probabilities to be used for speech recognition. A straightforward method to carry out this transformation was reported in [25] in the case of Artificial Neural Networks. The idea consists of taking the network outputs $g(\mathbf{x})$, that are considered as estimates of class-posteriors, and transforming them into class-conditional probabilities using Bayes' rule (see equation 4.4). The class-priors $p(k_i)$ can be obtained from the training material by counting the examples of each class k_i (i.e. counting phone occurrences). However, as pointed out in [53], class-priors can be adjusted during the test phase to compensate for training data with class probabilities that are not representative of actual use or test conditions.

$$p(\mathbf{x}|k_i) = \frac{p(k_i|\mathbf{x})p(\mathbf{x})}{p(k_i)} \tag{4.4}$$

For the particular case of speech decoding, the term $p(\mathbf{x})$ is normally omitted. The reason is that speech decoding consists in the comparison of competing sequences of words that start and end at the same time positions so the accumulated probability of the sequence of observations $p(\mathbf{x}_1^t)$ is the same for all of the word sequences at any given time t. Thus, expression 4.4 can be simplified into expression 4.5, which combined with expression 4.1 can be used to calculate the emission probabilities of each of the HMM-states.

$$p(\mathbf{x}|k_i) \propto \frac{p(k_i|\mathbf{x})}{p(k_i)} \tag{4.5}$$

4.2.2 HMMs topology

Once the procedure of calculating emission probabilities for HMM-states has been clarified, it is necessary to select the topology of the HMMs that will be used during the recognition process. State-of-the-art ASR systems doing acoustic modeling at the phone-level (as it is the case of the ASR system under consideration) typically utilize HMMs with a number of states ranging from three to five. A larger number of states is expected to model the temporal structure of a speech unit better, however it typically requires more training data to reliably estimate the increased number of parameters and will produce a slower decoding. In the other hand a too small number

of states may not model adequately the temporal structure of a speech unit.

In the case of SVMs, and especially if pairwise classifiers are the strategy selected for dealing with the multiclass classification, an increase in the number of HMM-states has a critical impact in the number of classifiers that need to be trained. In particular, this number grows quadratically with the number of HMM-states, while the number of samples available to train the classifiers only decreases linearly. Recall from section 2.3.1 that, under the one-vs-one scheme it is necessary to train $K(K-1)/2$ pairwise classifiers to solve a multiclass classification problem comprising K classes. In this case, the number of classes K corresponds to the sum of HMM-states across all the HMM-models, and can be expressed as $K = PS$, where P is the number of phonetic classes[11] and S is the number of states at each HMM. The number of classifiers that would need to be trained for different HMM-topologies is shown in table 4.1. As can be seen, the difference in the number of classifiers needed for different topologies is substantial. Additionally, figure 4.2 shows the pairwise classifiers needed to separate the HMM-states of two phone-models according to three different HMM topologies. Where the HMM-states s_i are represented as circles and the pairwise classifiers by bidirectional arrows.

After doing some informal experiments, one and three-state topologies were selected to build the HMM-models. In those experiments it was observed that, although training a considerably higher number of classifiers is expected to require considerably more training time, those classifiers are trained on a smaller number of samples (a training sample either is aligned to a HMM-state or another) and, given that SVMs require a training time that grows superlinearly with the number of samples, the training of the whole set of classifiers is even faster. In the other hand, it was observed that, the larger the number of pairwise classifiers used to separate the HMM-states of two classes, the larger the overall summation of support vectors and Lagrange multipliers resulting from the training of the classifiers. This has a negative impact in the calculation of emission probabilities and thus in the decoding time [12].

Based on those conclusions, three-state HMM-models seem to be a very reasonable option. It is a compromise between trainability and decoding time. However, in order to confirm the conclusions that, remember, where drawn from informal experiments, a one-state topology was also

[11] Note that phones are the unit selected for building the recognition system.

Number of HMM-states	Classifiers needed	Classifiers needed ($N = 52$)
One	$K(K-1)/2$	1326
Three	$3K(3K-1)/2$	12090
Five	$5K(5K-1)/2$	67340

Table 4.1: *Number of pairwise SVM classifiers resulting from different HMM-topologies.*

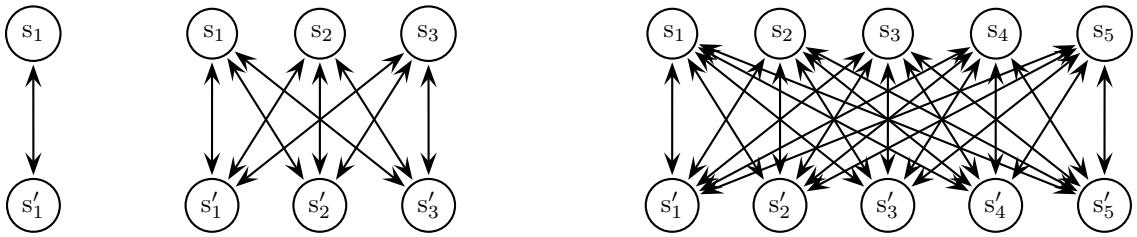

Figure 4.2: *Pairwise classifiers (represented by bidirectional arrows) needed to separate HMM-states (represented by circles) of different HMM topologies (from left to right, one, three and five-state topologies).*

considered and evaluated (the five-states topology was discarded due to the great computational cost that it would introduce in the Viterbi search). Speech recognition accuracies resulting from the use of these topologies will be shown in the next section.

4.2.3 Transition probabilities

Transition probabilities in a HMM represent the probability of moving from one HMM-state to all the allowed next HMM-states, which in a left-to-right HMM are the self-state and the state/states on its right (continuous speech recognition is based on concatenating HMMs). These probabilities can be efficiently estimated in a GMM/HMM system using the EM algorithm. In the case of a SVM/HMM system like the one described in previous sections, and for which the acoustic models are trained in a totally different fashion, it is not clear how to efficiently estimate the transition probabilities. Hybrid SVM/HMM approaches like the one described in [35] make use of transition probabilities obtained from the training of a GMM/HMM system, however such approach requires the training of two systems.

Nonetheless, several works exist in the literature showing that transition probabilities have

[12]It may appear that too much emphasis has been put in training/decoding time related issues, however, as it will be analyzed in section **??**, it is believed that this is one of the main shortcomings in the applicability of SVMs to speech processing.

a marginal impact on recognition performance. In [54] it is mentioned that the state transition probabilities have practically no effect on recognition performance. In [55], it is observed that it is not the state transition probabilities that force the HMM to find the correct segmentation of an observation sequence, but rather it is the emission probabilities that handle this. For these reason, and given that some state-of-the-art systems actually ignore transition probabilities, they have been removed from the SVM/HMM system proposed in this chapter and only a minimum duration of three time-frames per HMM has been imposed to control the phones' duration. In the case of a three-state HMM this constraint is implicit in the model topology, while for a one-state HMM the constraint must be set explicitly in the decoder.

4.3 System's evaluation

With the intention of evaluating the recognition accuracy of the SVM/HMM decoder described in previous sections, three different speech decoding systems have been used:

- A SVM/HMM system for which emission probabilities are computed as described in previous sections and transition probabilities have been removed. This system does not make use of context information for training the acoustic models[13]. Two variants of this system have been trained, one using one-state HMMs and another one using three-state HMMs.

- A conventional GMM/HMM speech recognition system has been trained using Sonic [41] that makes use of triphone-based acoustic modeling. This system have been used with the intention of showing the word accuracy of a state-of-the-art HMM making use of context dependency.

- Given that the available version of Sonic did not allow for the use of monophones (instead of triphones) during decoding, and in order to build a comparable system, a monophones system was also trained on the same corpora using HTK [56].

All these systems were trained in the same corpora and use a three-state HMM topology (a one-state HMM topology was also trained for the SVM/HMM system).

[13] A context dependent SVM/HMM system based on this one will be introduced in chapter 6

4.3.1 Experimental setup

The speech corpus used for the evaluation is the CU Read and Summarized Story Corpus [57]. This corpus is composed exclusively of children read speech, and it has been selected because, due to the great variability of children's speech, this is a especially challenging task for which state-of-the-art systems still failed to yield a good recognition performance. Thus, there is a larger margin for improvement.

From that corpus, only speech belonging to first graders (6-7 years old students for a total of 72 speakers) has been selected, and then partitioned into a training set containing 3 hours of audio and a test set of about 1 hour of audio. For every speech utterance contained in the training set, 39-dimensional feature vectors, consisting of 12 Mel Frequency Cepstral Coefficients and energy plus first and second order derivatives, have been extracted. The total number of feature vectors extracted is about 1144000[14]. Using the Sonic Speech Recognizer [41] we carried out a forced alignment to obtain state-level labels for each of the feature vectors. Note that this is the only connection of the SVM/HMM system proposed with a conventional GMM/HMM system. For training the SVM/HMM system, 52 three-state HMMs are used (corresponding to each of the phonetic labels used in the alignment).

For training the SVM classifiers a radial basis function (RBF) kernel is used, the optimal values of the error penalty parameter C and the kernel parameter γ are estimated independently for each classifier using a diluted subset of the training set. The parameters estimation is carried out doing a grid-search process using 5-fold cross validation. The library used for the training is LibSVM [58].

For decoding purposes an ad-hoc library it has been implemented that allows the evaluation of the decision function associated to each SVM classifier on such a way that kernel evaluations are reused across classifier's evaluations (see section 7.2 for a clarification on this issue).

[14]The reason why only about 3 hours of audio from the corpus were selected for the training is that training SVMs on datasets larger than about one million of samples results computationally extremely expensive.

4.3.2 Classifiers accuracy

This experiment has been carried out to evaluate the classification accuracy of the SVM pairwise classifiers trained for both, the one-state and the three-state HMM topologies. The number of classifiers trained for each of the topologies is shown in table 4.1. The classification accuracy has been measured on the test set described in the experimental setup. Classification results are depicted in figure 4.3, in which it is depicted the models distribution over the classification accuracy. Since the number of classifiers resulting from both (one-state and three-state) HMM architectures is different (much higher in the case of three-state HMMs), percentage values have been taken so the area below both curves is the same and they can be directly compared. Looking at the right-shift of the three-state HMM classifiers' curve, it is clear that those classifiers are considerable more accurate. This is not surprising since it is well known that the temporal structure of a phone is divided in at least three differentiated regions. In the next section it will be studied up to what extent this difference in classification accuracy translates into a better recognition accuracy.

An interesting detail is that, on average, the number of support vectors of pairwise classifiers is inversely proportional to their classification accuracy. This is shown in figure 4.4 in the case of pairwise classifiers trained for the three-states HMMs architecture. This is not surprising since, typically, a low classification accuracy is obtained as a consequence of separating very overlapping sets of feature vectors, separation for which many support vectors are needed. The interesting point is that, while high-accuracy classifiers are also fast to evaluate (have a reduced number of support vectors[15]), low-accuracy classifiers not oly produce unsatisfactory results but are slow to evaluate (have a considerable number of support vectors). For example, the pairwise classifier trained to separate samples from the central HMM-state of the phonetic classes AA and AO consist of 9785 support vectors (what represents 55% of the training samples) while its accuracy is only 72%. In the other hand the pairwise classifier trained for the very separable central HMM-state of AA and S, presents a classification accuracy of almost 100% while having only 812 support vectors (less than 3% of the training samples).

[15]Note that, since the distribution of the data across different classes depends on the task, this is an expected value.

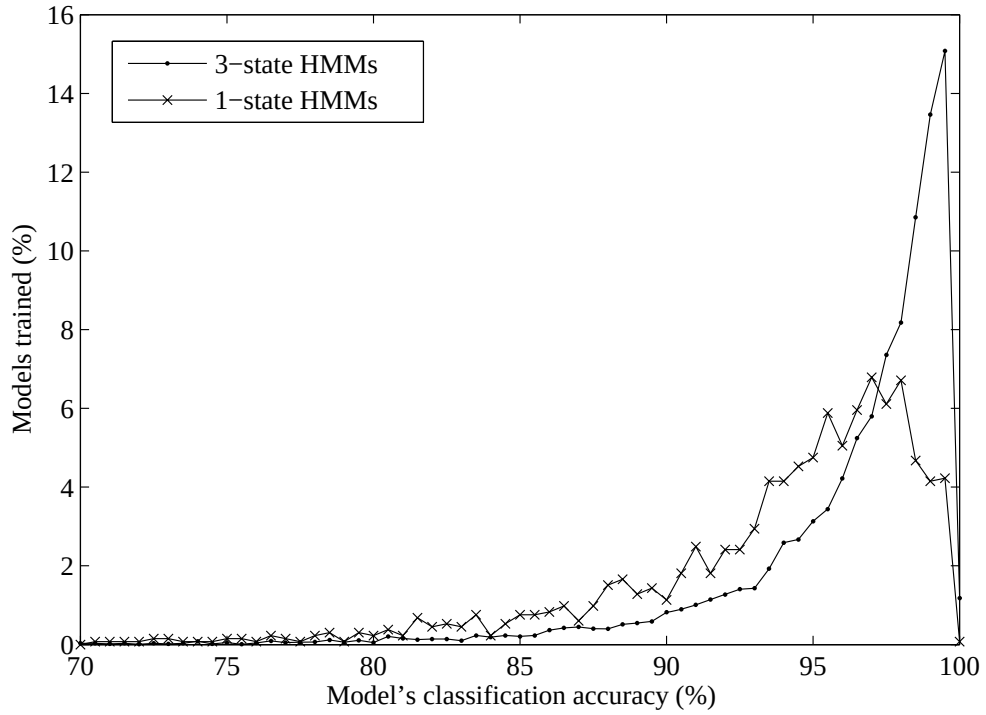

Figure 4.3: *Classification accuracy comparison between pairwise classifiers for one-state and three-state HMMs.*

4.3.3 Classifiers accuracy and position of the HMM-states in the model topology

Is has been considered interesting to analyze the separability between HMM-states depending on their position in the HMM-model to which they belong. For this reason, the classification accuracy of the pairwise classifiers obtained in the previous experiment has been averaged across the 6 possible pairwise combinations of HMM-states. Table 4.2 shows the results. Two observations have been made:

- Pairwise classifiers for which at least one of the classes is a central HMM-state present the highest accuracy. In particular, pairwise classifiers trained to separate central states of different HMMs are the most accurate.
- Pairwise classifiers trained to separate initial and final HMM-states are the less accurate ones.

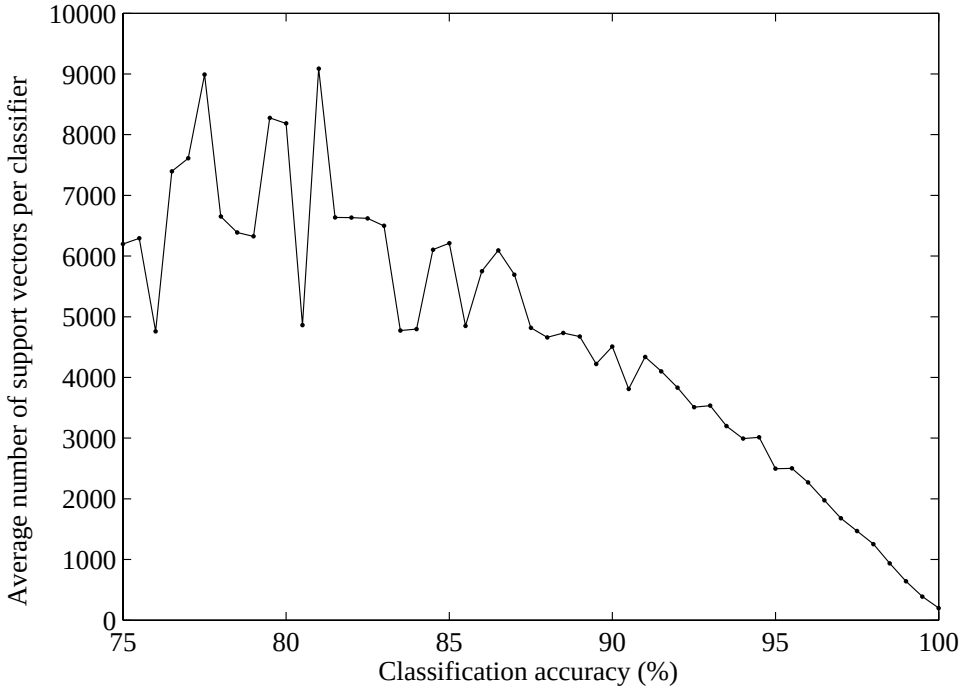

Figure 4.4: *Relationship between the classification accuracy of a classifier and the number of support vectors for three-state HMMs.*

These observations are not surprising given that the central state of a HMM is the most stable one and feature vectors aligned to it are the most homogeneous ones across different realizations of the same phonetic class. For this reason, feature vectors aligned to the central state of an HMM are expected to be more easily separable from feature vectors aligned to other states. In the other hand features frames aligned with the initial and final state of a HMM are more influenced by the realization of the preceding and seceding phones so are less homogeneous.

4.3.4 Recognition accuracy

In this experiment it has been compared the word error rate (WER) of the three ASR systems described at the beginning of section 4.3. The systems have been compared using a uniform language model (all the words in the lexicon receive the same probability) and a trigram. As can be observed

HMM-state	first state	second state	third state
first state	0.9564	0.9697	0.9614
second state		0.9712	0.9689
third state			0.9596

Table 4.2: *Averaged classification accuracy of classifiers trained to separate HMM-states at different positions.*

in table 4.3 the SVM/HMM system significantly outperforms the GMM/HMM monophone system, however the difference respect to the system making use of contextual information is considerable. Next chapters will be focused on closing this gap by incorporating more information into the training of the acoustic models.

System	uniform	trigram
GMM/HMM monophones (HTK)	41.47%	
SVM/HMM monophones (baseline)	44.46%	55.45%
GMM/HMM triphones (Sonic)	55.07%	67.53%

Table 4.3: *Word accuracy of the proposed SVM/HMM system respect to a comparable monophone GMM/HMM system and a triphone system.*

4.4 Conclusions

In this chapter a SVM/HMM based continuous speech recognition system has been introduced and experimentally evaluated. Advantages of the proposed system over other applications of SVMs to speech recognition have been presented and justified. In addition to that, implementation details of the system have been provided. The proposed system makes use of SVMs as probabilistic estimators of emission probabilities and attains a superior word error rate than a comparable GMM/HMM system. However, despite these appealing results the application of SVMs to speech recognition under the proposed framework still needs to face many challenges. In the next chapters it will be explored a mechanism to incorporate contextual information into into the training. In addition to that several scalability issues of the proposed system will be discussed.

Chapter 5

Implicit State-Tying

When training pairwise classifiers as described in the previous chapter, it was found that those trained to discriminate between some of the HMM-states of similar phonetic classes present a very low cross-validation accuracy. For example, pairwise classifiers trained to discriminate between the first state of AA, AH or AO, or pairwise classifiers trained to discriminate between the last state of the closures BD, DD, GD, KD, PD or TD, present very poor discriminative performance. This is not surprising since phones belonging to the same broad phonetic class are expected to be significantly overlapping and thus pairwise classifiers trained to discriminate between their states are expected to perform poorly. In these cases, a SVM trained for probability estimation would ideally produce probability values close to 0.5, thus reflecting appropriately the low accuracy of the model. However, it has been observed that the probability estimates obtained from pairwise classifiers trained to discriminate between very overlapping classes are very unreliable, which significantly deteriorates the overall emission probability computation process expressed described in section 4.2.1. The reason is that, usually, when a feature frame belonging to a broad phonetic class is examined, pairwise classifiers trained to discriminate between phonetic classes belonging to that broad class and the remaining classes produce probabilities very close to one. Ultimately, this causes that the posterior probability of the best scoring class is strongly determined by the probabilities obtained from the pairwise classifiers that discriminate between the classes included in the same broad class.

5.1 Tying procedure

5.1.1 Introduction

To cope with this problem, it has been proposed to remove from the posterior probability computation process expressed in 4.1 (and thus from the emission probability computation process) those pairwise classifiers that are expected to produce unreliable posterior estimates. For example, while SVMs trained to discriminate between the second or third HMM-state of AA and AY present a cross-validation accuracy above 92%, the SVM trained to discriminate between the first state of AA and AY presents a cross-validation accuracy below 70%. This suggests removing this classifier from the posterior computation process of the first state of both classes AA and AY while still using the pairwise classifiers trained to discriminate between the second and third state. Since the emission probability of the first HMM-state of AA and AY will be calculated excluding the pairwise classifier trained for both, this procedure can be considered an implicit tying of states.

Figure 5.1 shows the pairwise classifiers that need to be trained to separate HMM-states from phonetic classes AA and AY according to the standard pairwise scheme. Circles in the figure represent HMM-states (for example, the node AA_1 represents the first state of the HMM model of AA) while each bidirectional arrow denotes a pairwise classifier trained to separate the HMM-states pointed to. The corresponding scheme once the first state of AA and AY are tied, can be observed in figure 5.2. Note that, despite that only a pair of states have been tied (AA_1 and AY_1), five pairwise classifiers have been removed, the reason is that, if two states are tied, they can be considered (at least locally, and this will be clarified later on) as belonging to the same class, and, it is not convenient to train pairwise classifiers to distinguish between HMM-states of the same class.

Once the process of tying HMM-states locally (i.e. between two phonetic classes) has been described, the question is that whether HMM-states tied locally should also be tied globally. For example, if AA_1 and AY_1 are tied and considered the same HMM-state when training pairwise classifiers between phonetic classes AA and AY, should these HMM-states also be merged globally for training pairwise classifiers with HMM-states of other classes? The answer to that question is not clear, however it has been observed that, while two HMM-states, let's say s_1 and s_2 can be

very similar to each other, each of both can present a very different degree of similarity respect to a third HMM-state s_3. For this reason, while it may be beneficial to tie s_1 and s_2 when calculating the emission probabilities, it may be harmful to assume that s_1 and s_2 are exactly the same class and only a single classifier needs to be trained to separate s_1 and s_2 from s_3. For this reason, the tying procedure only takes place at the decoding stage and not at the training stage.

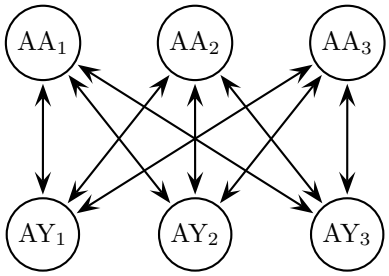

Figure 5.1: *Pairwise classifiers (represented by bidirectional arrows) trained to separate HMM-states from the phonetic classes AA and AY before the tying process.*

5.1.2 Tying method

The tying method proposed consists of doing 5-fold cross-validation on the training data (only 1/5 of the training samples are used in order to speed up the process [16]) and obtaining a cross-validation accuracy for each of the $K(K-1)/2$ pairwise classifiers. Using CV_{ij} as the cross-validation accuracy of the pairwise classifier trained to discriminate the classes k_i and k_j, expression 5.1 substitutes 4.1 in the calculation of posterior probabilities. Where ν represents the minimum cross-validation accuracy required to include a pairwise classifier in the computation.

$$p(k_i|\mathbf{x}) = \left[\sum_{\substack{j=1, j\neq i, \\ CV_{ij} \geq \nu}}^{K} \frac{1}{p(k_i|k_j \; or \; k_i, \mathbf{x})} - (K-2) \right]^{-1} \tag{5.1}$$

[16] An important performance consideration is that the cross-validation accuracy of a classifier can be obtained from the cross-validation process carried out to estimate the parameters of the probabilistic output (A and B parameters in 2.20). Thus, no extra computations are needed. This is possible because computing the classification accuracy does not require probability estimates.

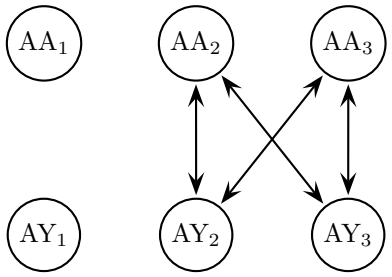

Figure 5.2: *Pairwise classifiers (represented by bidirectional arrows) trained to separate HMM-states from the phonetic classes AA and AY after the tying process.*

Finally, it is important to note that when building each one of the 5 subsets of training samples, contiguous feature vectors aligned with the same HMM-state must be placed into the same subset. Otherwise the correlation of adjacent spectral features will produce unreliable cross-validation results.

5.2 Experiments

An experiment has been carried out to evaluate the effect of the implicit state-tying method proposed in the previous section on the recognition accuracy. Initially, a decoding process is done in which the complete set of pairwise classifiers (12090) is used to compute the emission probabilities (this is equivalent to the SVM/HMM system evaluated in section 4.3.4). Then, a 1:5 diluted set of the training data (only one out of five feature vectors is used) is used to calculate the cross-validation accuracy of each of the pairwise classifiers (The same cross-validation process is used to estimate the parameters for the probabilistic output). A threshold ν is utilized to select the subset of pairwise classifiers that will be used to calculate the emission probabilities following expressions 5.1 and 4.5.

Fig. 5.3 shows a comparison between the word accuracy of both systems. It can be observed that as the value ν increases the word accuracy gets better with respect to the baseline. However once the value of ν goes beyond a certain value (about 92% in the figure) the accuracy of the system deteriorates significantly. This is not surprising since an elevated value of ν causes many pairwise classifiers with an acceptable discriminative performance to be removed from the emission

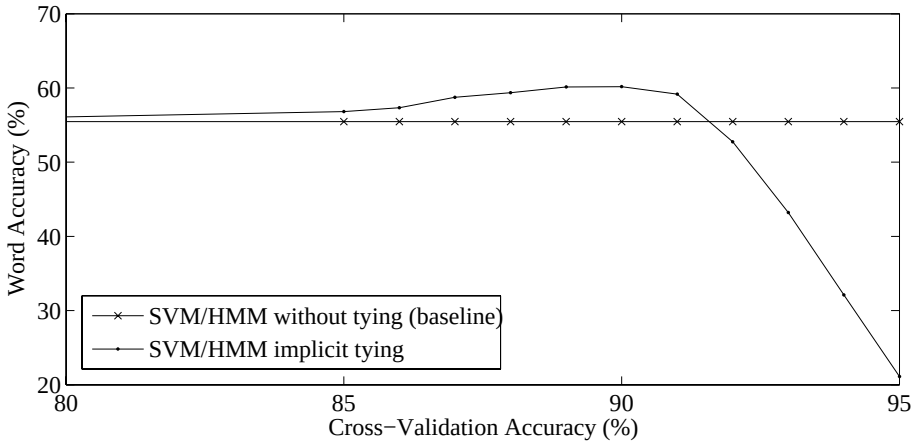

Figure 5.3: *Word accuracy for different values of ν.*

probability computation process. It is important to note that, for example, for a value of ν equal to 90, about 19% of the pairwise classifiers may be removed from the training process. So the tying approach not only improves accuracy but speeds up the training process. Table 1 shows the word accuracy of the system with respect to the baseline and GMM/HMM systems trained with the same corpus. It can be seen that the proposed state-tying system clearly outperforms comparable monophone systems. Two probabilistic language models have been used for the evaluation, a uniform language model (i.e. all the words in the lexicon are considered equally likely) and a trigram.

System	uniform	trigram
GMM/HMM monophones (HTK)	41.47%	
SVM/HMM monophones (baseline)	44.46%	55.45%
SVM/HMM monophones (state-tying)	48.71%	60.19%
GMM/HMM triphones (Sonic)	55.07%	67.53%

Table 5.1: *Word accuracy of the proposed state-tying SVM/HMM system respect to comparable monophone systems and a triphone system.*

5.3 Conclusions

Summarizing, removing unreliable pairwise classifiers from the emission probabilities computation process presents the following advantages:

- Word accuracy improvement: The tying procedure proposed has shown to help in the reduction of the WER, indicating that training classifiers to separate very similar classes is not recommended. However, although under a monophone perspective those classes are very close, those classes are potentially very separable if context information would be used for training the classifiers. Next chapter is dedicated to explore this issue.

- Faster decoding: fewer classifiers need to be evaluated at the frame level during the Viterbi search, with the consequent reduction in computation.

- Faster training: after doing the cross-validation process those classifiers with an accuracy below the threshold do not need to be trained. Recall that the cross-validation accuracy is calculated on a diluted set so it is considerably faster than training the final classifiers on the whole dataset.

- No extra computation is needed: the classification accuracy of each classifier can be obtained from the standard cross-validation process required to compute the probabilistic output.

Finally note that the removal of a certain number of classifiers from the emission probabilities computation process without compromising accuracy is in part possible thanks to the fact that training 9 classifiers (recall figure 4.2) to separate each pair of phonetic classes, although presenting several advantages as discussed in 4.2.2, is somehow redundant.

Chapter 6

Context Dependency

6.1 Introduction

In the last chapters a SVM/HMM continuous speech recognition system was proposed and experimentally evaluated showing very promising results in terms of WER respect to comparable GMM/HMM systems. In addition to that, in the last chapter it was found that some HMM-states are very confusable and training SVM-classifiers to separate between them does not produce satisfactory results. This is not surprising given that the system was based on monophones and it is very well known that the realization of a phone is strongly influenced by the preceding and successive phones. It is for this reason why most state-of-the-art systems make use of contextual information. However it is not clear how such kind of information can be incorporated into the proposed system or, in general, into a SVM-based speech decoding system. In particular there are two main issues that need to be covered when dealing with contextual features and discriminative classifiers:

- Automatic identification of the contextual classes to model.
- Modeling of the contextual classes.

This chapter is dedicated to explore the utilization of contextual information for acoustic modeling using SVMs and shed some light on those two issues.

6.2 Motivation of context dependency

There are several reasons that make words the most natural unit for speech recognition, first, determining the spoken sequence of words is the ultimate objective and, second, word models are able to capture the within-word contextual effects so the inter-word variability of phones can be effectively modeled. Consequently, when the vocabulary is limited and the training material is sufficient, words usually show the best performance in comparison with subword units such as syllables, or phones. However, in many real applications, for which the amount of training data is limited and the vocabulary is large, word models may not be adequately trained given that there may be not enough examples of each word. Another major inconvenience of a word-based system is its rigidity, i.e. whenever a new word needs to be incorporated into the vocabulary the system has to be retrained.

For these reasons, most state-of-the-art ASR systems use phones instead of words as the modeling unit. For large or middle-size vocabulary tasks, the number of phones is typically very reduced in comparison with the number of different words, which results in much fewer parameters to estimate and consequently in a more efficient training. However, a phone realization is strongly affected by its preceding and successive phones, so training context-dependent models is much needed in order to attain satisfactory recognition accuracy [59]. Context-dependent phone models are a compromise between specificity (allow to model the phonetic co articulation) and trainability (the total number of context-dependent models can be adjusted to meet the size of the available training material). When the number of words is in relatively large (in the order of several thousands) training context-dependent phone models requires the estimation of considerably fewer parameters[17].

When phone-level context-dependent training is carried out, the speech representation unit selected is typically the triphone (although other units like diphones or quinphones have been successfully applied, which feasibility mostly depends on the available training material), which consists of a phone with its corresponding preceding and successive phones. A triphone is typically denoted as ph_l-ph+ph_r, where ph_l and ph_r are, respectively, the phones acting as left and right

[17] If for example triphones (see next paragraph) are used for modeling the context, typically only few thousand triphone models need to be estimated, of course this may vary from one corpora to another.

context of the basephone ph.

The number of different triphones that may occur in a given language, like, for example, English, is extremely high so training a different HMM-model for each of them would impose a great demand of training material. This demand would be further accentuated if cross-word triphones are considered (i.e. if context dependency is modeled not only within word boundaries but across word boundaries). To cope with this problem, ASR systems have used over time a number of techniques focused on improving the trainability of the context-dependent models. These techniques usually fall in one of the three categories that are summarized next[18]:

- Parameters sharing: this technique, also known as tying parameters, consists of sharing some of the parameters of different HMMs. In the case of triphones, it's been observed that many phonetic classes have the same or similar effect in the realization of a phone when preceding or following it, thus it is possible to cluster them into broader classes for which a context dependent model is trained. The procedure of clustering different triphones corresponding to the same basephone is called triphone-clustering and can be carried out in a number of ways. Next section is dedicated to outline some of the most widespread techniques for clustering triphones in a conventional GMM/HMM system. Typically, parameters are shared across different triphones belonging to the same basephone, however it is possible to tie parameters across triphones of a different basephone [60].

  The benefits of tying are the following:

  - For a given training set, reducing the overall number of parameters leads to a more robust estimation.
  - A reduced number of parameters allows a faster training and has less storage requirements.

  However, parameter sharing has a clear limitation and it is that each tying represents an information loss, thus, it is of major importance to carefully select the parameters to be tied so this loss is minimal. For example, in [61] it is concluded that under a certain occurrence threshold (35 occurrences) state-tying results in a splitting of rarely seen training material

[18] State-of-the-art ASR systems typically use a combination of those techniques for improved recognition accuracy.

and leads to less robust, modeling. The tying of parameters can be carried out at multiple levels, for example in [62] it is shown how parameters of context-dependent phone models can be tied in a hierarchical fashion, from tying a whole HMM to tying states, mixtures, gaussians or even means of gaussians.

- Backing-off: this technique consists of training specific models (triphone models) only when enough training data is available for an effective estimation. Otherwise the models are backed off to a less specific model for which enough training data is available. For example moving from a triphone to its corresponding left or right biphone or even to the basephone model if not enough data is available for a more specific training. As it will be shown later on, backing-off is widely used in bottom-up clustering approaches.

- Smoothing: smoothing techniques aim to increase the robustness of the trained models and consist in smoothing the parameters of more specific models with those of less specific models, which have been trained on more data and, hence, are more reliable. Examples of smoothing techniques can be found in [63] and [64],

6.3 A review of triphone-clustering techniques

Triphone-clustering is one of the most widespread techniques for parameter sharing. It consists of finding subsets of triphones (composed of one ore more elements) that are phonetically similar (hence, can be trained together) and at the same time have a sufficient number of occurrences in the training material to be robustly trained. As it was mentioned in the previous section, there exists a hierarchy between a basephone and its corresponding biphones and triphones, this hierarchy goes from less specificity (the basephone) to more specificity (the biphones and then the triphones). Clustering techniques are designed to take advantage of this hierarchical structure to find subsets of triphones that are suitable for being trained together. Attending to how the hierarchical structure (tree) is traced in the clustering, these techniques can be divided into two categories:

- Bottom-up: bottom-up approaches [65] initially create a cluster for every state[19] in the training material and progressively merge clusters, for example selecting at each iteration the pair of clusters which when combined form the smallest resultant cluster, until a compact set of

clusters is obtained. The stopping criteria for the merging process is commonly based on two parameters: a maximum cluster size and a minimum number of clusters. While bottom-up techniques are very widespread they present a well known shortcoming, they do not deal with unseen triphones, i.e. triphones for which there are no occurrences in the training material and therefore no cluster was built for them. This problem becomes particularly serious when cross-word context modeling is used and the system is evaluated on large corpora, in this situation the number of unseen triphones is usually significant. These situations are commonly handled during decoding in two different ways:

1. Backing-off to a less specific model like a biphone or a monophone. The problem is that this less-specific models also need to be trained [61], additionally the accuracy of the less specific model deteriorates the recognition performance.
2. Mapping the unseen triphone to the closest cluster in the training data by means of a similarity measure [66].

- Top-down: these approaches [67] aim to cope with the problem of unseen triphones by starting from a single cluster containing all the states and iteratively splitting the clusters using a set of binary rules until a certain stopping condition is met. This procedure is typically carried out using a binary decision tree for which, once the process ends, terminal nodes become clusters of states that will be modeled together. At the beginning the root node contains all the states to be clustered, at each iteration a leaf of the tree is split using a set of binary phonetic rules that is built using phonetic knowledge[20], the rule that yields the best log-likelihood gain is applied and the cluster is split into a left node containing the states satisfying the rule and a right node with the remaining ones. A terminal node is no longer split when either the log-likelihood gain[21] of the best rule applicable or the number of training samples in the node falls below a given threshold.

  Some of the advantages of the top-down procedure over the bottom-up are listed next:

[19] Typically, the clustering is carried out across states of triphones belonging to the same basephone and that are in the same position of the model topology.

[20] Other sources of information have also been successfully applied, like syllable structure [68] or speaking rate [69].

[21] Note that the log-likelihood, which is the most popular objective function used for clustering, always grows with each split given that splits are trained and evaluated on the same training data.

- It is possible to find a cluster, and thus an acoustic model, for every triphone observed during decoding by simply tracing down the decision tree. Thus, no backing-off or mapping techniques are needed.
- Prior knowledge extracted from different complementary sources can be incorporated into the clustering procedure in a very elegant fashion.
- It allows to easily control the minimum number of samples of each cluster so the HMMs can be robustly trained.

Nevertheless, the main shortcoming of this method is that, typically, the thresholds used as the stopping criteria need to be estimated empirically. In general, finding the right balance between the total number of parameters and the training data used to train them is a key factor in any clustering technique.

6.4 Context dependent training in a SVM/HMM system

It is clear that context dependent training is mandatory to properly modeling phonetic coarticulation in a phone-based speech recognizer. Although many techniques have proven excellent performance to do such a modeling in a GMM/HMM system (see previous section) or an ANN/HMM hybrid system [70] [71], to date, no such technique exists in the case of SVM-based speech recognition. This section is focused on finding an effective technique to train context-dependency for a stand-alone SVM/HMM system.

State-of-the-art GMM/HMM systems do context-dependent acoustic modeling by first, applying a triphone-clustering technique like the ones described in previous section and, second, training an acoustic HMM for every cluster of triphones. In the case of an ANN/HMM hybrid system, context modeling is typically carried out in two steps:

1. A GMM/HMM system is used to determine the phonetic contexts of a given phone label that will be modeled separately.

2. A series of neural networks are trained to compute the class-posterior probabilities of each of the phone's clusters. This can be done, for example, doing a factorization of probabilities like

in [70] or training a context specific network for each phone and each context class identified [71].

When the acoustic modeling is carried out using SVM classifiers as described in chapter 4, the necessity of modeling context is quite clear. In fact, table 4.2 shows that the classification accuracy of pairwise classifiers used to separate edge-states (i.e. states at the edges of the HMM-topology) is significantly worse than that of the rest of classifiers. This confirms that the initial and final regions of a phone are strongly influenced by the preceding and successive phones, so making use of contextual information is much needed. As previously introduced, triphones are a very successful approach for doing such a context modeling, however, none of the conventional triphone clustering methods existing in the literature (which are designed for GMM/HMM systems) can be directly applied in the case of SVMs pairwise context modeling. The reason is that SVM models work on finding decision boundaries between classes and it is not clear what those classes would be under a conventional clustering procedure nor how the clustering procedure itself would be carried out.

Nonetheless, it would be desirable to design and implement a SVM-based triphone clustering procedure that is inspired on traditional clustering methods and thus takes advantage of all the nice properties that such procedures show. In order to get to that, several issues need to be addressed, they are the following:

1. A technique is needed to identify the context classes for which SVM classifiers will be trained. If triphones are the basic context unit, a triphone-clustering technique is needed to identify the clusters of triphones for which SVMs will be trained. The idea consists of finding clusters of triphones that are more compact and can potentially be more easily separable from the other classes than the original context-independent set of models.

2. A multiclass-classification strategy must be selected for training the context-dependent models of each cluster of triphones.

3. A procedure must be defined to compute the emission probabilities using the resulting context-dependent classifiers.

Techniques to deal with these issues will be covered along the next sections.

6.4.1 Introduction

In chapter 4 a SVM/HMM system making use of context-independent models was described and evaluated. In such a system, samples belonging to a pair of classes k_i and k_j (HMM-states) are separated using a pairwise classifier trained only with samples of both classes. These pairwise classifiers (which outputs can be combined to compute emission probabilities) are the basic unit for acoustic modeling and, thus, are the target for the application of context modeling. Having the sets of triphones P and Q containing triphones of the HMM-states k_i and k_j respectively, a straightforward mechanism for training context dependency would be breaking P and Q into smaller non-overlapping subsets of triphones $p_i \in P = \{p_1, \ldots, p_{L_P}\}$ and $q_j \in Q = \{q_1, \ldots, q_{L_Q}\}$ for which $L_P \times L_Q$ pairwise classifiers (p_i, q_j) can be trained. These context-dependent pairwise classifiers (p_i, q_j) would be trained to separate more compact classes (each class is composed only by phonetically close triphones) so can potentially attain superior classification accuracy than the context independent classifier. Other properties that make this idea appealing, are listed next:

- The set of clusters into which the triphones of class k_i are split can be different depending on the class k_j for which pairwise classifiers are being trained. This way, the resulting sets of clusters could be designed to optimize the separability between samples of classes k_i and k_j. In its turn, in order to separate triphones from classes k_i and a third class k_l, a new set of clusters would be obtained from both classes to maximize the separability.

- While the number of pairwise classifiers (p_i, q_j) that need to be trained $(L_P \times L_Q)$ in the context independent case grows quadratically with the number of clusters L_P and L_Q of classes k_i and k_j, those classifiers are trained on much smaller datasets than the context independent one, so the overall training time may be even smaller. Recall from chapter 2 that the training time of SVMs scales superlinearly with the number of training samples. For example, one-vs-one pairwise classifiers can be trained faster than one-versus-rest classifiers on the same training material. Additionally, context dependent classifiers are trained on potentially more separable data so the decision function resulting from the training is expected to comprise only a reduced number of support vectors respect to the total number of training samples, which is always a desirable property.

- Only a percentage of all the pairwise classifiers trained are expected to be benefited from context dependent training. The explanation for this is the following: given a HMM-state, the pairwise classifiers that need to be evaluated to calculate its emission probability can be roughly divided into two broad categories:

  - Pairwise classifiers trained to separate phonetically close classes: for example, if the emission probability of the first state of the phonetic class AA is being calculated, pairwise classifiers that fall into this category would be, among others, those trained to separate the first state of AA from the first state of AO, AH or AY.
  - Pairwise classifiers trained to separate phonetically far classes: are those that were trained to separate easily separable classes like the first state of AA and the first state of T, V, or P.

  Although many pairwise classifiers fall in the middle of both categories, looking at figure 4.3 where the test accuracy of the pairwise SVM classifiers trained was shown, it is clear that a big number of pairwise classifier present a discriminative accuracy that is already very high and, thus, can hardly be improved. Those classifiers are not expected to be benefited from context dependent training so simple context-independent classifiers will be enough to perform a reliable discrimination. Note that, while contextual information is extremely useful for resolving ambiguity, there are a number of pairs of HMM-states that are not confusable, like for example UW and T. A pairwise modeling scheme is able to take full advantage of this issue.

This idea is depicted in figure 6.1, where triphones belonging to the phonetic classes k_1 and k_2 are split into three disjoint clusters of triphones. Following the clustering procedure, a context independent pairwise classifier (denoted by a bidirectional arrow) is transformed into nine pairwise context-dependent classifiers, each of them separating a different pair of clusters.

6.4.2 Preliminary experiments

In order to evaluate the viability of the previously described scheme for context dependent training, a small-scale experiment was conducted using the experimental set-up described in section 4.3.1.

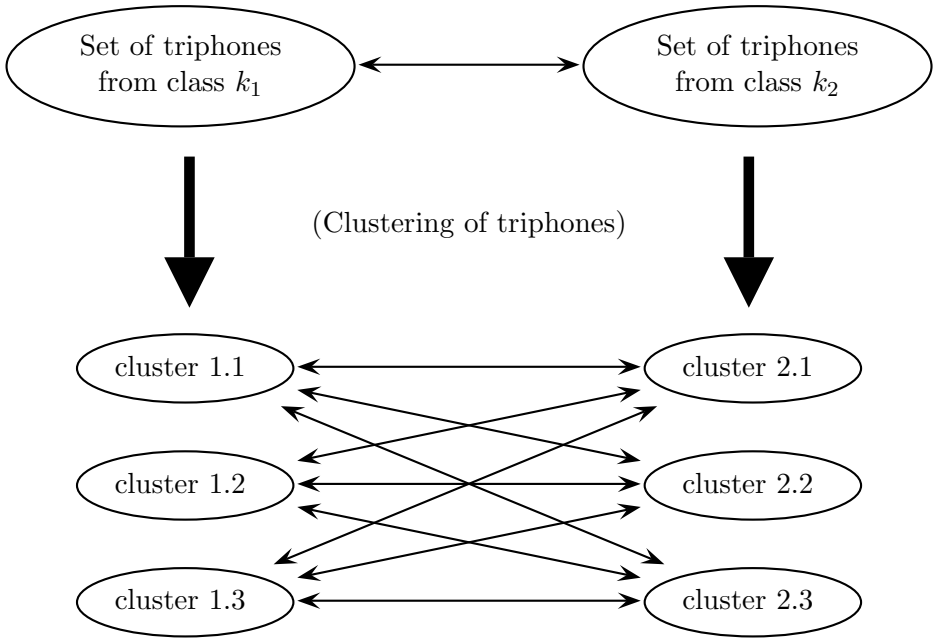

Figure 6.1: *Transition from a context-independent classifier to context-dependent classifiers through a process of triphone clustering.*

In this experiment a context independent (CI) pairwise classifier was trained for the first state of phonetic classes AA and AO. Additionally, with the intention of training context-dependent (CD) classifiers, triphones from both states were clustered by hand using phonetic similarity rules. Given that the first state of a HMM is usually more influenced by the left context, only that context was taken into account for the hand-made clustering. Clusters were created so the number of training samples in each of them was potentially enough to robustly train the classifiers. Finally, pairwise classifiers were trained for all the combinations of clusters of both classes. In order to compare the classification accuracy of the CD classifier respect to the CI ones, all the trained classifiers were evaluated with feature vectors from the test partition of the corpora as described in the experimental setup.

Table 6.1 shows the hand-made clusters for triphones belonging to the first state of classes AA and AO. For each cluster it is shown the left context of the triphones clustered together and the occupancy. It can be observed that the number of samples per cluster is quite balanced and is always above 100. This means that every pairwise classifier will be trained on a relatively balanced

Clusters of AA			Clusters of AO		
Cluster	Left contexts	# samples	Cluster	Left contexts	# samples
1	SIL	737	1	S	623
2	N,NG	616	2	W	602
3	F	563	3	SIL	570
4	T	469	4	F	296
5	G,K	386	5	M,N	281
6	R	384	6	Y	249
7	D	352	7	AXR,R	223
8	W	249	8	HH,P,TS,V,Z	197
9	CH,JH,TS	224	9	TH	184
10	M	214	10	AY,EY,IY,OW,UW	174
11	P	201	11	DD,KD,TD	150
12	DD,KD,TD	189	12	D,T	129
13	IY,Y	182	13	G,K	109
14	HH	158	14	L	104
15	B	152			
16	AW,AXR,OW,Z	131			
17	S,SH	129			
18	L	117			
19	UW	113			
20	AY,EY	105			

Table 6.1: *Hand-made triphone clusters for the first HMM-state of phonetic classes AA and AO.*

dataset of at least 200 samples. This number of samples, which corresponds to only two seconds of speech, may appear to be too few to robustly train a SVM classifier. However, it will be shown that, if the classes are separable, very few number of training samples are required to attain satisfactory results.

For each pair of clusters in table 6.1 a pairwise SVM classifier was trained using a Gaussian RBF function. Then, a force-alignment was carried out between the speech utterances in the test corpora and their corresponding transcriptions. Finally, samples aligned with the first HMM-state of AA and AO were clustered[22] into the clusters of table 6.1 and used to evaluate the classification accuracy of the corresponding pairwise classifiers (each test sample is evaluated on a a number of classifiers equal to the number of clusters of the complementary phonetic class). The resulting classification accuracy is shown in table 6.2.

Looking at table 6.2, it can be observed that the classification accuracy of the context

[22]It should be mentioned that, given that the clustering procedure was made by hand, a small number of triphones in the test set have no cluster to be clustered into so have no CD pairwise classifier to be evaluated with and thus were not used for evaluation purposes.

rows / cols = clusters of the first HMM-state of AA / AO.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.95	0.99	0.59	0.99	0.91	1.00	0.97	0.87	0.94	0.92	0.79	0.94	0.97	0.94
2	0.85	0.94	0.85	0.99	0.85	0.95	0.87	0.87	0.89	0.84	0.84	0.91	0.94	0.91
3	0.94	0.87	0.94	0.95	0.89	0.97	0.81	0.85	0.90	0.96	0.90	0.90	0.88	0.91
4	0.71	0.91	0.98	0.97	0.96	0.85	0.68	0.88	0.73	0.91	0.88	0.75	0.87	0.78
5	0.84	0.92	0.95	0.99	0.99	0.96	0.81	0.83	0.85	0.97	0.91	0.79	0.77	0.92
6	0.88	0.86	0.96	0.90	0.83	0.95	0.60	0.86	0.87	0.92	0.93	0.79	0.91	0.87
7	0.72	0.97	0.95	0.96	0.87	0.74	0.91	0.87	0.79	0.91	0.83	0.44	0.83	0.88
8	0.99	0.75	0.98	0.95	0.84	0.99	0.96	0.97	0.99	1.00	0.93	0.96	0.95	0.72
9	0.80	0.92	0.92	0.91	0.85	0.83	0.82	0.85	0.84	0.84	0.70	0.73	0.90	0.83
10	0.85	0.94	0.68	0.86	0.63	0.96	0.87	0.74	0.92	0.68	0.67	0.88	0.91	0.75
11	0.91	0.93	0.87	0.94	0.90	0.95	0.83	0.66	0.86	0.91	0.85	0.89	0.86	0.85
12	0.94	0.99	0.79	0.99	0.83	0.95	0.92	0.78	0.91	0.83	0.60	0.80	0.95	0.90
13	0.94	0.98	0.97	1.00	0.90	0.78	0.85	0.88	0.93	0.74	0.66	0.78	0.98	0.90
14	0.93	0.92	0.84	0.91	0.82	0.90	0.85	0.74	0.85	0.84	0.80	0.85	0.59	0.73
15	0.87	0.84	0.86	0.82	0.78	0.91	0.69	0.73	0.89	0.78	0.82	0.84	0.89	0.84
16	0.92	0.95	0.72	0.95	0.76	0.90	0.76	0.72	0.81	0.54	0.59	0.79	0.87	0.77
17	0.83	0.94	0.69	0.97	0.75	0.89	0.63	0.65	0.85	0.45	0.59	0.83	0.80	0.93
18	0.93	0.89	0.80	0.97	0.76	0.99	0.83	0.77	0.91	0.68	0.73	0.89	0.89	0.73
19	0.96	0.95	0.81	0.93	0.84	0.86	0.84	0.85	0.87	0.71	0.70	0.89	0.86	0.91
20	0.96	0.96	0.81	0.96	0.84	0.83	0.85	0.82	0.90	0.62	0.68	0.90	0.99	0.78

Table 6.2: *Classification accuracy over the test set of the context-dependent classifiers trained for every pair of clusters of the first HMM-state of classes AA and AO.*

independent classifiers is very uneven. While some classifiers seem to be very accurate, others present a very low classification accuracy that, in some cases, is very close to 0.5 (zero discriminative power). The reason is that classifiers trained to separate samples of AA and AO with similar or equal left context cant find a satisfactory decision function because the classes are just not separable. In the other hand, classifiers trained to separate samples with very different left context usually find very good decision functions, resulting in a very good generalization.

For example, the pairwise classifier trained to separate the 8th cluster of class AA from the 10th cluster of class AO has an accuracy of 1.00 (100% of the test samples are correctly classified), this is not surprising since the 8th cluster of AA contains triphones with left context {W} and the 10th of AO the left contexts {AY, EY, IY, OW, UW} that have a very different effect in the realization of the phones AA and AO respectively and, thus, can be easily separated. In the other hand, the classifier that separates the twelfth cluster of AA from the eleventh cluster of AO has an accuracy of 0.60, again, this expected because both clusters are composed of triphones with the same left contexts {DD, KD, TD} and, thus, can be barely separated.

Nevertheless, as it is shown in table 6.3, the accuracy of the context dependent classifiers is substantially superior (on average) than that of the context independent classifier. While only 68% of the test samples are correctly classified using the context independent classifier, 87%[23] of the test samples are correctly classified using the context dependent ones.

In addition to that, on average, each context dependent pairwise classifier is comprised of fewer support vectors than the context independent one. In particular, 6871 samples out of the 9562 total training samples become support vectors in the CD case, which represents a 72%. In the CI case, only an average of 204 support vectors result from training on datasets of 562 samples on average, which represents only a 36%. Once again, it is shown that when the percentage of support vectors respect to the total number of training samples is very high, it is usually an indicator that either not enough data is available to find a good decision function or the samples of both classes are not easily separable. Anyhow, this results in poor classification accuracy.

[23]Making use of the whole set of pairwise CI classifiers, 32830 and 25460 sample evaluations were carried out using the test samples from triphones of classes AA and AO. Out of those, 28645 and 21947 evaluations produced correct classifications, so the class-dependent accuracy was 0.87 and 0.86 respectively.

[24]The number between parenthesis indicates the number of unique support vectors, that is always a lower bound of the total number of training samples and, as will be discussed in chapter 7, is a very good indicator of the time required to evaluate a set of SVMs trained on overlapping data.

	Context indep. classifier	Context dep. classifiers
Classification accuracy	0.68	0.87
# training samples (total)	9562	9562
# training samples (per model)	9562(5671,3891)	562(284,278)
# support vectors (total)	6871	57185(8637)[24]
# support vectors (per model)	6871	204

Table 6.3: *Classification accuracy for context independent and context dependent classifiers separating the first HMM-state of phonetic classes AA and AO.*

Despite that the small size of the experiment conducted prevent from drawing well-founded conclusions, results are very encouraging. For this reason, and in order to build a context dependent ASR system under this paradigm, an automated triphone clustering technique has been proposed. This technique will be thoroughly described in the next sections.

6.4.3 SVM-based triphone clustering technique

In previous section, a technique for doing context-modeling using SVM pairwise classifiers was proposed and partially evaluated showing promising results. However, the clustering procedure in which that technique lied, was hand-made. In this section a completely automated technique for clustering triphones will be detailed. This technique will allow to perform triphone clustering for any pair of HMM-states so the context-dependent training can be fully automated.

Assuming a top-down approach (which allows an effective handling of unseen triphones), triphone-clustering techniques start with a big cluster containing all the triphones of a given basephone and iteratively split the clusters at the lower level of the tree into disjoint subclusters so that a function (for example, likelihood gain) is maximized on each split. If pairwise SVM classifiers are used for acoustic modeling, triphones from two HMM-states need to be clustered in such a way that the separability between samples of both classes is maximized. Therefore, it seems to make sense to simultaneously cluster the triphones of both classes using a couple of decision trees that are expanded alternatively. As in typical top-down approaches, a cluster can be split by evaluating a set of phonetic rules and applying the one that maximizes a certain function. While for a conventional GMM/HMM system this function is usually the log-likelihood, for a SVM/HMM system the function could be the classification accuracy between terminal nodes in both trees. A straightfor-

ward mechanism for computing the classification accuracy between terminal nodes is doing f-fold cross-validation. This way, every time a cluster of triphones (terminal node in one of the trees) is selected for split, the rule that maximizes the cross-validation classification accuracy (i.e. the separability) between the resulting subnodes and the terminal nodes of the complementary tree, is selected. A cluster is no longer split when either its occupancy or the "separability gain" resulting from applying the best available rule falls below a given threshold.

The mechanism for calculating the so called "separability gain" and related concept definitions required for the clustering process will be described in the next section.

Splitting gain and termination criterion

This section is dedicated to define a series of concepts needed for the application of the clustering technique. Given a tree node n, its separability respect to the leave nodes in the complementary tree is defined as expressed in 6.1. Where M is the number of terminal nodes m_i in the complementary tree of n, c_{n,m_i} is the number of correctly classified samples of node n using the pairwise classifier trained to separate samples from nodes n and m_i, and s_a is the number of samples of node a. The separability of a node is a value in the interval $[0, 1]$ and it is obtained from the classification accuracy of pairwise classifiers trained to separate that node from all the terminal nodes in the complementary tree. A f-fold cross-validation process needs to be carried out to calculate the classification accuracy of each of the M pairwise classifiers. During those processes, each sample from node n is evaluated by M classifiers while each sample from the complementary clusters is evaluated by only one classifier, for this reason the accumulated number of correctly classified samples of node n needs to be normalized by M. Intuitively, this separability value corresponds to the average classification accuracy of the classifiers that separate node n from the nodes m_i in the other tree.

$$\text{separability}(n) = \frac{\frac{1}{M}\sum_{i=0}^{M} c_{n,m_i} + \sum_{i=0}^{M} c_{m_i,n}}{s_n + \sum_{i=0}^{M} s_{m_i}} \tag{6.1}$$

Figure 6.2 shows the pairwise classifiers that need to be trained to compute the separability between a terminal node p_2 and the terminal nodes in the complementary tree $\{q_1, q_2, q_3\}$. Whenever

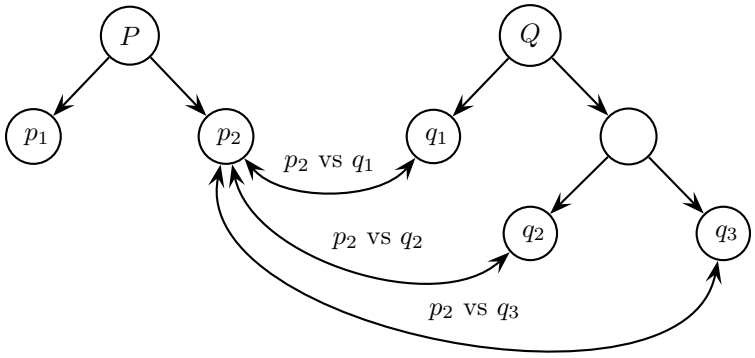

Figure 6.2: *Pairwise classifiers (denoted by bidirectional arrows) needed to compute the separability of triphones in node p_2 from triphones in the terminal nodes of the complementary decision tree $\{q_1, q_2, q_3\}$.*

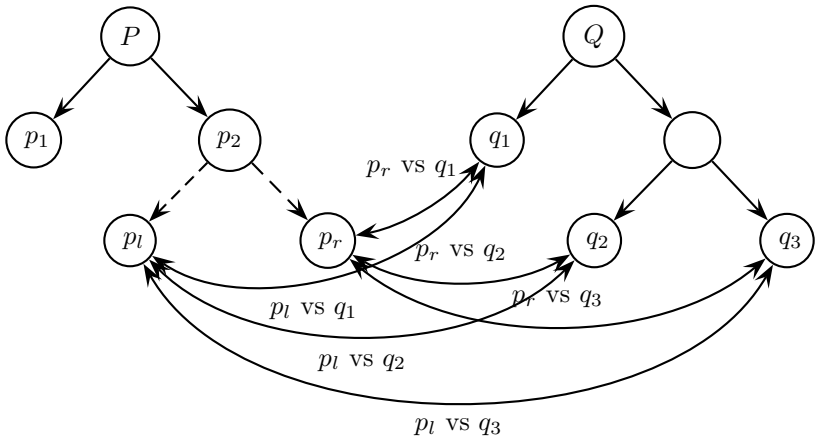

Figure 6.3: *Pairwise classifiers (denoted by bidirectional arrows) needed to compute the split-separability value resulting from the split of the node p_2 into subnodes p_l and p_r.*

a node n is selected for split, a split-separability value is computed as expressed in 6.2 for each of the applicable phonetic rules r. To calculate this value, pairwise classifiers need to be trained to separate samples from the resulting left and right subnodes (n_{left} and n_{right}) and the terminal nodes in the complementary tree m_i.

Figure 6.3 shows the pairwise classifiers needed to compute the split-separability value resulting from the split of the node p into subnodes p_l and p_r. In this case, the subnodes p_l and p_r, need to be separated from the terminal nodes $\{q_1, q_2, q_3\}$ in the complementary tree.

$$\text{split-separability}(n,r) = \frac{\frac{1}{M}\sum_{i=0}^{M}\left(c_{n_{left},m_i} + c_{n_{right},m_i}\right) + \frac{1}{2}\sum_{i=0}^{M}\left(c_{m_i,n_{left}} + c_{m_i,n_{right}}\right)}{s_n + \sum_{i=0}^{M} s_{m_i}} \tag{6.2}$$

Once a split-separability value has been calculated for any applicable rule r, the rule that yields the highest separability gain is applied[25]. The separability gain is defined as in 6.3 and it is the ratio between the separability of the left and right nodes resulting from applying the rule and the separability of the original node. Hence, the separability gain is bigger than 1 if the resulting subnodes n_{left} and n_{right} are more separable than the original node n.

$$\text{separability-gain}(n,r) = \frac{\text{split-separability}(n,r)}{\text{separability}(n)} \tag{6.3}$$

The separability of the left and right nodes resulting from the split is calculated as expressed in 6.4 and 6.5 respectively.

$$\text{separability}(n_{right}) = \frac{\frac{1}{M}\sum_{i=0}^{M} c_{n_{right},m_i} + \sum_{i=0}^{M} c_{m_i,n_{right}}}{s_n + \sum_{i=0}^{M} s_{m_i}} \tag{6.4}$$

$$\text{separability}(n_{left}) = \frac{\frac{1}{M}\sum_{i=0}^{M} c_{n_{left},m_i} + \sum_{i=0}^{M} c_{m_i,n_{left}}}{s_n + \sum_{i=0}^{M} s_{m_i}} \tag{6.5}$$

Construction of the decision trees

Once the clustering technique has been outlined and some related concepts have been defined, it is time to describe step by step the clustering algorithm. As previously mentioned, the procedure consists in building a couple of decision trees which nodes are expanded alternatively using the

[25] In practice, the best rule is applied only if its separability gain is greater than a certain threshold.

HMM-state	context to which the phonetic rules are applied
first state	left context
central state	full context
third state	right context

Table 6.4: *Contexts to which the phonetic rules are applied*

separability between the terminal nodes in both trees (as previously defined) as the function to maximize. The algorithm, which must be repeated for each pair of phonetic classes k_1 and k_2 (HMM-states), is detailed next:

1. Two empty lists $L_{active(k_1)}$ and $L_{active(k_2)}$ are created to keep the tree nodes (clusters of triphones) that are considered for split at each iteration.

2. Two empty lists $L_{final(k_1)}$ and $L_{final(k_2)}$ are created to keep the tree nodes that are considered final, i.e. will not be further split. Triphones contained into those nodes will be clustered together.

3. Two root nodes n_1 and n_2 are created and the available training triphones of classes k_1 and k_2 are placed together in the corresponding node. Along with the triphones, the set of phonetic rules that will be used for splitting are placed at each of the root nodes according to table 6.4 and the position of the HMM-states to which classes k_1 and k_2 refer[26]. Additionally, a separability value between the whole set of triphones of classes k_1 and k_2 is calculated using expression 6.1 and doing cross-validation[27]. Note that this separability value corresponds to the classification-accuracy of the context independent classifier and that for the root nodes separability(k_1) = separability(k_2). The resulting separability value is stored at each of the root nodes and will be used as a reference for the evaluation of potential splits. Every time a split is made, the new separability value for each of the subnodes is computed and stored. Finally, the root nodes n_1 and n_2 are inserted into the lists $L_{active(k_1)}$ and $L_{active(k_2)}$ respectively.

[26] In order to reduce the number of rules that need to be evaluated for a split and, hence, accelerating the clustering procedure, full context is not always considered. For example, according to table 6.4, only the left context is considered for clustering triphones relative to the initial state of a HMM. This is coherent with the assumption that the initial region of a phone is mostly influenced by the preceding phone.

[27] Details about how the cross-validation is done can be found in section 6.4.3

4. At each iteration, a tree node n is selected for split. With the intention of training SVM classifiers on datasets as much balanced as possible, the node selected is the one containing the greatest number of training samples. However, while in the first iteration that node is selected considering nodes in lists $L_{active(k_1)}$ and $L_{active(k_2)}$, in subsequent iterations only the nodes contained in one of the lists (which changes on each iteration) are considered. This constraint is applied in order to guarantee that consecutive splits take place in complementary decision trees. The reason is that it's been observed that splits in one decision tree favor splits in the complementary decision tree and sometimes the only way of successfully splitting a node is by splitting first some of the nodes of the complementary tree. If the node selected for split contains a number of samples below a given threshold μ_s the node is moved to the list $L_{final(k)}$, where k is the class of the node. In case both lists $L_{active(k_1)}$ and $L_{active(k_2)}$ are empty, the algorithm proceeds to step 7.

5. In order to select the best split of node n into subnodes n_{left} and n_{right}, all the phonetic rules R_n available at node n are evaluated. For each phonetic rule r in R_n, the following steps are carried out:

   (a) Using the rule r (which, according to table 6.4, can be applied either to the left, right or full context) the set of triphones in node n is split into two disjoint subsets of triphones that are placed into nodes n_{left} and n_{right} so triphones that meet the rule are placed in n_{left} and the remaining ones are placed in n_{right}. In case that the number of samples in either n_{left} or n_{right} falls below a given threshold μ_s, the rule is discarded and the algorithm proceeds to the next rule. This threshold is used to establish a lower bound on the number of samples used for training a SVM classifier so the classifier is trained robustly.

   (b) Being $\{m_i\}, i = 1, \ldots, M$ the terminal nodes[28] in the decision tree of the complementary class of node n, a classification accuracy value is obtained from classifiers separating the following pairs of nodes $(n_{left}, m_1), \ldots, (n_{left}, m_M)$ and $(n_{right}, m_1), \ldots, (n_{right}, m_M)$ doing f-fold cross-validation. Once these classification-accuracy values are obtained, the split-separability resulting from the split of the node n into subnodes n_{left} and n_{right}

using the rule r can be calculated using the equation 6.2.

(c) Finally, a separability-gain value separability-gain(n, r) resulting from the application of the rule r to split the node n is calculated using the expression 6.3.

6. Once all the available rules are evaluated, the rule r_{best} that produces the highest split-separability is applied if separability-gain$(n, r_{best}) \geq \mu_{acc}$. In case the rule is applied, the node n belonging to class k is removed from the list $L_{active(k)}$ and the subnodes n_{left} and n_{right} resulting from the split using r_{best} are inserted into that list, having each of them a separability value calculated using expressions 6.4 and 6.5 respectively. The whole set of rules in node n is copied into nodes n_{left} and n_{right} with the sole exception of the rule r_{best}. If there are no rules to copy, the subnodes n_{left} and n_{right} are inserted into the list $L_{final(k)}$ instead of $L_{active(k)}$. In case no rule has been applied, the node n is inserted into the list $L_{final(k)}$ and discarded for further splitting attempts. At this point the selected node has been processed and the algorithm proceeds to step 4.

7. At this point all the splits have been carried out and the final nodes in lists $L_{final(k_1)}$ and $L_{final(k_2)}$ are the resulting clusters of triphones for which pairwise classifiers must be trained. The decision tree can be recovered by tracing the tree down from the root and storing the binary rules applied for each split.

Finally, note that the parameter μ_s (minimum number of samples in a node so it can be considered for splitting) used as stopping criterion must be set so there are enough training samples to train the classifiers. It's been observed experimentally that a reasonable number is around 100. In its turn, the parameter μ_{acc}, used to control the minimum separability gain of a split, must be set experimentally.

Procedure for obtaining the cross-validation accuracy

An important drawback of using the classification accuracy of pairwise classifiers as the basis for the computation of the separability-gain during the splitting procedure is that it is computationally very expensive. In fact, doing a cross-validation procedure on the whole set of samples of each pair

[28] Note that the terminal nodes of a class k are $L_{active(k)} \cup L_{final(k)}$.

of terminal nodes would be intractable. For this reason, and in order to control the clustering time, only a diluted set of training samples is used for training and evaluating the context-dependent SVM classifiers required to perform the cross-validation. This exploits the property of SVMs according which the training time can be kept low on small datasets.

Two parameters are used to control the training and evaluation time during the cross-validation:

- c_t: this parameter represents the maximum number of samples that are selected from each cluster of triphones for training the classifiers. However, the classifiers are always trained on balanced datasets so this parameter is only an upper-bound of the total number of samples taken from each class. The reason is that the clustering procedure previously described, often requires to compute the separability of very unbalanced clusters of triphones, which usually produces decision boundaries biased toward the class with more representatives. Another strategy would be using different error-penalties for each class.

- c_e: this parameter represents the maximum number of samples that are selected from each cluster of triphones for evaluation purposes. While the training is always carried out on balanced datasets, the evaluation can be done in very unbalanced datasets. Note that this parameter does not depend on the value of c_t. While c_t needs to be kept low given that the training time grows superlinearly with the number of samples, the value of c_e can be set more flexibly because only impacts the evaluation time linearly.

In addition to the use of these thresholds, folds are built subject to two constraints[29]:

- Samples from each triphone are shared out among different folds in approximately equal proportion. The intention is creating folds that are as much representative of the whole set of clustered triphones as possible.

- Samples aligned with the same phone in the training material are kept in the same fold. Otherwise, given the strong correlation between adjacent feature vectors, the cross-validation results would be unreliable.

[29]In order to allow that these constraints can be met, the minimum number of training samples for each class c_t must be set appropriately.

Optimizations of the SVM-based clutering technique.

In order to evaluate the previously proposed SVM-based clustering technique, some informal experiments were conducted. The experiments consisted of clustering triphones belonging to a few pairs of HMM-states and evaluate the separability of the resulting clusters. While the proposed technique seemed to produce an effective clustering of triphones it showed to be computationally extremely expensive. This is not surprising since for each split a set of rules needs to be evaluated requiring each of them the training and evaluation of a number of SVM classifiers which increases as the number of terminal nodes in both trees increases. For example, if a node n is selected for split into nodes n_{left} and n_{right} and there are 10 terminal nodes (clusters of triphones) in the complementary tree, according to the clustering technique described and having 40 binary rules applicable at node n, the total number of pairwise cross-validation processes needed is $10 \times 2 \times 40 = 800$. If a 5-fold cross-validation is used, 4000 classifiers need to be trained and evaluated just for computing the split-separability of the node n. Although these classifiers can be trained and evaluated on diluted sets of samples (as described in previous section), which requires only fractions of a second, the whole clustering procedure remains still very slow.

In order to reduce the computational cost involved in building the decision trees, a modification has been introduced with the intention of reusing rule evaluations. The idea consists of selecting multiple rules for splitting a node at each iteration. This way, after the set of applicable rules are evaluated for splitting a cluster of triphones n (node in the tree) into clusters n_{left} and n_{right} and the best rule r_{best} (the one that yields the best separability-gain) is selected, the rest of evaluated rules are analyzed and used for further splitting if it is convenient. To introduce this optimization, the clustering procedure described previously needs to be modified at step 6, which becomes:

6. Once all the available rules are evaluated, the rule r_{best} that produces the highest split-separability is applied if separability-gain$(n, r_{best}) \geq \mu_{acc}$. In case the rule is applied, the node n belonging to class k is removed from the list $L_{active(k)}$ and the subnode n_{left} resulting from the split using r_{best} is inserted into that list, while the subnode n_{right} will be used for further splitting attempts. The separability value of both subnodes is calculated using expressions 6.4 and 6.5 respectively. The set of rules in node n is copied into nodes n_{left}

and n_{right} with the exception of r_{best}. In case no rule has been applied, the node n is inserted into the list $L_{final(k)}$ and discarded for further splitting attempts. In this case the algorithm proceeds to step 4.

At this point, a set of rules $R' = R - \{r_{best}\}$ containing all the rules R evaluated for splitting the node n (except the best rule r_{best} already used to produce n_{left} and n_{right}) with their corresponding subclusters and separability-gain, is available resulting from step 5. These rules are sorted in descending order of separability-gain and will be analyzed for splitting the subnode n_{right}. Triphones in this subnode are moved to a temporal cluster of triphones (i.e. they are not inserted in any decision tree) called n_{rest} that represent the triphones in n that can be potentially reclustered within this iteration. Additionally, triphones in n_{left} are copied into another temporal cluster called $n_{clustered}$ that represents the triphones that are already clustered within this iteration. Note that both temporal clusters are disjoint $n_{clustered} \cap n_{rest} = \emptyset$ and that $n_{clustered} \cup n_{rest} = n$ is always guaranteed.

For any rule $r_i \in R'$ that splits the node n into subnodes $n_{left,i}$ and $n_{right,i}$ the following checks are made:

- Triphones in $n_{left,i}$ (i.e. triphones that satisfy the rule r_i) do not overlap with the set of triphones in n_{left}. This constraint is set to ensure that $n_{left,i}$ and n_{left} are disjoint sets of triphones and thus can coexist as terminal nodes.
- The separability-gain resulting of applying r_i on node n is positive. This constraint is set to avoid the reevaluation of unpromising phonetic rules.

In case those requirements are met, the rule r_i is evaluated for splitting n_{rest} into subnodes $n_{left,i}$ and $n_{right'}$ with the advantage that the classification accuracy resulting from separating the node $n_{left,i}$ from the terminal nodes in the complementary tree $\{m_i\}, i = 1, \ldots, M$ was already computed in step 5. Then, it is only necessary to compute the classification accuracy resulting from separating the subnode $n_{right'}$ from the $\{m_i\}$. Note that this is the central point of the optimization because it allows to do additional splits at half the computational cost of evaluating a single rule. Once those classification accuracies are computed, if separability-gain$(n_{right}, r_i) \geq \mu_{acc}$ then the r_i is applied and the node $n_{left,i}$ is inserted into the list

$L_{active(k)}$ having a separability value computed using 6.4. The last step prior to the evaluation of the next r_i consists of adding triphones in $n_{left,i}$ to $n_{clustered}$ and replacing the triphones in n_{rest} by the triphones in $n_{right'}$. Note that $n_{rest} = n_{left,i} \cup n_{right'}$.

Finally , once all the rules in R' are examined, the node n_{rest} (which contains a fewer number of triphones respect to the original n_{right} in case that more than one rule has been applied) is inserted into the list $L_{active(k)}$. At this point the selected node has been processed and the algorithm proceeds to step 4.

Note that if only the best rule r_{best} is applied (i.e. a single split is carried out), this procedure is completely equivalent to the non-optimized one. Additionally, it is important to observe that the order in which the rules $r_i \in R'$ are evaluated for further splitting does not guarantee that the rule yielding the best separability-gain over n_{rest} is applied first. Nevertheless, this procedure is expected to produce a satisfactory clustering while consuming considerable less computational resources than the original approach.

6.5 Emission probability calculation of a HMM-state using pairwise context-dependent SVM classifiers

Once the context-dependent SVM pairwise classifiers have been trained for each pair of HMM-states, the calculation of the emission probability of a HMM-state given the feature vector $\mathbf{x}$ at time t can be carried out in a similar fashion to that used for context-independent classifiers. Recall from section 4.2.1 that the emission probability of a context-independent HMM-state k_i is calculated by evaluating all the pairwise classifiers trained to separate k_i from the remaining HMM-states (classes) $k_j, j \neq i$ and then combining the probabilistic values obtained ($p(k_i|k_j \; or \; k_i, \mathbf{x})$) using expression 4.1. When context-dependent classifiers are used, if a is a cluster of triphones belonging to the HMM-state k_i for which the emission probability needs to be computed, the class-posterior probability of cluster a can be computed as follows:

$$p(a \in k_i|\mathbf{x}) = \left[\sum_{j=1, j\neq i}^{K} \frac{1}{p(a|k_j \; or \; a, \mathbf{x})} - (K-2) \right]^{-1} \tag{6.6}$$

Where the pairwise class-posterior of cluster a and a HMM-state $k_j, j \neq i$ (compounded by a set of clusters $\{b_l\} \in k_j, l = 1, \ldots, n$) can be computed as follows:

$$p(a|k_j \; or \; a, \mathbf{x}) = \frac{\prod_{b_l=1}^{n} p(a|b_l \; or \; a, \mathbf{x})}{\prod_{b_l=1}^{n} p(a|b_l \; or \; a, \mathbf{x}) + \prod_{b_l=1}^{n} (1 - p(a|b_l \; or \; a, \mathbf{x}))} \tag{6.7}$$

Finally the emission probability of the feature vector $\mathbf{x}$ given the cluster of triphones a can be calculated dividing by the prior probability of the phone q to which the HMM-state k_i is relative:

$$p(\mathbf{x}|a) \propto \frac{p(a|\mathbf{x})}{p(q)} \tag{6.8}$$

Comparing this procedure for calculating the emission probabilities with the procedure described in section 4.2.1 for context-independent classifiers, the main difference is that, while only one classifier was needed to calculate the pairwise class posterior $p(k_i|k_j \; or \; k_i, \mathbf{x})$, n classifiers need to be evaluated now for calculating the analogous "cluster-posterior" $p(a|k_j \; or \; a, \mathbf{x})$. This is depicted in figure 6.4. However, although according to this procedure typically a greater number of classifiers needs to be evaluated[30], those classifiers are expected to be considerably smaller (i.e. are composed of considerable fewer support vectors) than the context-independent ones. This will be experimentally shown in the next section.

6.5.1 Context-dependent pairwise classifiers by example

This section is dedicated to make a preliminary evaluation of the previously proposed approach for training context-dependent pairwise classifiers. The experimental set-up is the same as in section 4.3.1 and the experiment consists of finding clusters of triphones for the phonetic classes EH and UH so the separability between them is maximized.

Table 6.7 shows the classification accuracy and the resulting number of support vectors of the 9 context-independent pairwise classifiers that separate HMM-states of classes EH and UH. Accuracy values between parentheses correspond to the positive and negative samples used in the

[30]It depends on the number of resulting clusters

Figure 6.4: *Pairwise classifiers (represented by bidirectional arrows) needed to calculate the pairwise class-posterior for context-independent classifiers (on the left) and context-dependent classifiers (on the right).*

	Classification accuracy			Number of support vectors		
HMM-state	UH(1)	UH(2)	UH(3)	UH(1)	UH(2)	UH(3)
EH(1)	0.96 (0.99,0.93)	0.95 (0.96,0.92)	0.95 (0.95,0.92)	1189	1018	1948
EH(2)	0.97 (0.97,0.97)	0.96 (0.96,0.95)	0.94 (0.94,0.89)	1219	1450	1869
EH(3)	0.98 (0.98,0.96)	0.95 (0.97,0.91)	0.91 (0.93,0.86)	732	1004	2591

Table 6.5: *Context-independent models for 3-state HMM modeling*

evaluation. Table 6.6 shows the number of clusters and support vectors resulting from the proposed clustering technique. Finally, table 6.7 shows the classification accuracy of the context-independent pairwise classifiers that separate between the corresponding clusters in table 6.6.

As can be seen in the tables the clustering procedure is able to find clusters with a pairwise separability that is considerable superior to the separability of the corresponding context-independent classifier. An interesting point is that this procedure presents some similarities respect to the part-versus-part strategy proposed in [12]. The main difference is that, in this case,

	Number of classifiers trained			#support vectors: average (total)		
HMM-state	UH(1)	UH(2)	UH(3)	UH(1)	UH(2)	UH(3)
EH(1)	10 (2x5)	15 (3x5)	20 (4x5)	184 (1843)	161 (2418)	180 (3601)
EH(2)	6 (3x1)	15 (3x5)	20 (5x4)	290 (870)	143 (2151)	210 (4213)
EH(3)	2 (2x1)	15 (3x5)	45 (9x5)	396 (792)	163 (2619)	150 (6749)

Table 6.6: *Analysis of the clustering and training results of context-dependent pairwise classifiers for the phonetic classes EH and UH.*

	Classification accuracy (averaged across triphones)		
HMM-state	UH(1)	UH(2)	UH(3)
EH(1)	0.97 (0.98,0.97)	0.98 (0.99,0.98)	0.98 (0.99,0.97)
EH(2)	0.99 (0.99,0.99)	0.98 (0.99,0.98)	0.98 (0.99,0.98)
EH(3)	0.99 (0.99,0.99)	0.98 (0.99,0.98)	0.98 (0.98,0.98)

Table 6.7: *Context-dependent models for 3-state HMM modeling*

the disjoint subsets of training samples (clusters of triphones) are selected using prior knowledge (phonetic similarity rules) instead of randomly.

6.6 Conclusions

An automatic triphone clustering mechanism has been proposed that finds clusters of triphones which pairwise separability is maximized. The algorithm has been evaluated on a "toy experiment" that shows its potential. While the number of pairwise classifiers is increased, the context-dependent classifiers are trained on smaller datasets, are more accurate and present a fewer number of support vectors. Nevertheless, a much more detailed evaluation of the proposed technique would be required before drawing conclusions. Considering the amount of effort that performing such evaluation would require, further evaluations and enhancements of this technique as well as its full integration in a SVM/HMM context-dependent system are left for future work.

Chapter 7

Scalability

While the SVM-based speech recognition framework discussed in previous chapters has shown very promising results in terms of WER, the computation of emission probabilities using pairwise classifiers presents serious scalability issues. It is believed that the ability to properly overcome such scalability shortcomings will play a fundamental role in the success of the proposed framework. While the training time scalability appears to be the biggest concern when using nonlinear SVMs for classification tasks in which training on large datasets is required, in the particular case of speech decoding, in which all the acoustic models need to be loaded in main memory at once, the size of the SVM classifiers trained becomes also of major importance. Additionally, existing methods for obtaining class-posterior probability estimates in multi-class classification tasks, require the evaluation of all the classifiers trained (or at least a big percentage of them as it will be shown later on). In the case of speech decoding, which implies the separation of a big number of classes, the computational cost resulting from evaluating the whole set of classifiers at any time frame (about 10ms in most state-of-the-art systems) is extremely high so methods to address this problem are much needed.

This section is devoted to review different scalability problems that have been identified and propose several methods with the intention of minimizing them. The proposed methods have been incorporated into the SVM/HMM speech decoder described in previous sections and have been evaluated resulting in considerable performance gains.

7.1 Training time scalability

7.1.1 Introduction

SVMs that make use of non-linear kernels, and in particular Gaussian Radial Basis Function kernels, have shown the best performance in a number of pattern recognition tasks and are used in a wide variety of applications dealing with real-world data. According to previous work [22], this applies to the case of cepstral features classification what represents the core of the proposed SVM/HMM decoding system. However, as described in section 2.1 the training of a nonlinear SVM implies solving the following quadratic programming (QP) problem:

$$Q(\alpha) \equiv \sum_i \alpha_i - \frac{1}{2} \sum_i^n \sum_j^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{y}_i) \tag{7.1}$$

Where n is the number of training samples. Solving this quadratic optimization problem implies the computation of all the pairwise kernel evaluations $K(\mathbf{x}_i, \mathbf{y}_i)$ what has a computational complexity of $O(n^2)$. However, finding a solution to the problem may scale up to $O(n^3)$ and requires a storage space of $O(n^2)$, which for large datasets is computationally extremely expensive. Note that for the algorithm to perform at reasonable cost all the calculations must be carried out in main memory.

This is a very well know shortcoming of SVMs and several algorithms have been proposed over time to reduce the computational complexity of solving the QP problem:

- Low-rank approximations on the kernel matrix: these approximations can be obtained in a number of ways: using the Nystrom method [72], doing a greedy approximation [73], by means of sampling [74] or by matrix decompositions [75]. However, when training on very large datasets, the resulting rank of the kernel matrix may still be too high to be handled efficiently.

- Chunking: this technique was introduced in [10] and consists of optimizing the Lagrangian on an arbitrary subset of data. After the optimization, the set of nonzero Lagrange multipliers (α_i in equation 7.1) are retained while the other points in the subset are discarded. This procedure is iterated until the Karush-Kuhn-Tucker (KKT) conditions are met and thus the margin is maximized. However, the size of the subproblem tend to increase what, for large

datasets, implies solving a QP on an increasing number of samples.

- Decomposition methods: these methods, firstly introduced in [76] present similarities with the chunking methods and consist of breaking down the QP into a series of smaller subproblems and then use a numeric QP optimizer to solve each of them. However, for large datasets the size of these subproblems and consequently the computational cost of solving them may become very high.

- Sequential minimal optimization (SMO): this is a particular case of decomposition taken to the extreme, where the smallest possible optimization problem (containing only two variables) is solved at each step [77]. This technique has gain increasing popularity and it is used by state-of-the-art SVM libraries like LibSVM [78].

- Training SVMs on smaller datasets: this can be carried out, for example, selecting the input samples doing active learning [79] or decomposing the original dataset into several fixed-size subsets and training a different SVM for each of them, which outputs can be combined using a neural-network [80]. Note that the later procedure guarantees a training time linear with the number of samples by dividing the original dataset into fixed-size subsets. Clustering techniques have also been used, consisting of applying a similarity measure for grouping the training samples and then training a SVM with representatives of each cluster [81]. The main shortcomings of these techniques is that require an initial step of preprocessing the training samples and that some parameters are needed to control the samples selection, which best values need to be estimated empirically.

- Core Vector Machines: this recent approach [82] combines techniques from computational geometry with SVM training by reformulating the quadratic problem (QP) as a minimum enclosing ball (MEB) problem. This technique has shown time complexity asymptotically linear with the number of samples and space complexity independent of the number of samples. Experiments carried out on different datasets [82] showed that this technique produces classification accuracy comparable to state-of-the-art SVM training techniques while exhibiting considerable less training time for very large datasets and reducing the complexity (number of support vectors) of the classifiers produced.

Core Vector Machines has shown a remarkable performance compared to other existing techniques for training on large datasets, thus it would be interesting to explore the applicability of this technique to the case of cespstral features classification for speech processing. Note that, although this technique has been evaluated for several datasets [82], the nature of the features of those datasets was very different to the nature of cepstral features. Cepstral features classification is a especially difficult problem due to the usually large number of real-valued parameters of the feature vectors.

The remaining of this section includes a brief description of this technique and an experimental evaluation for cepstral features classification.

Core Vector Machines is based on reformulating the quadratic programming problem expressed in 7.1 as a minimum enclosing ball (MEB) problem for which a near optimal solution is obtained by means of an iterative $(1+\epsilon)$-approximation algorithm. The algorithm consists of maintaining a core-set S_t, which is a subset of the training samples $\mathcal{S}$ in the feature space corresponding to the kernel k, and its MEB (i.e. the smallest ball which contains all the points in $\mathcal{S}_t$) at each iteration t. The ball $B(c_t, R_t)$, where c_t and R_t are respectively the center and radius of the MEB at iteration t, is expanded at each iteration including a point falling outside $B(c_t, (1+\epsilon)R_t)$ into the core-set. The iterative process stops when all the points in S lie inside $B(c_t, (1+\epsilon)R_t)$. Once the $(1+\epsilon)$-approximate solution is found, the primal variables associated to the SVM can be recovered from $c = [\mathbf{w}'\ b\ \sqrt{C}\xi']$. The iterative algorithm introduced in [82] for solving the MEB problem involves the recomputation of the radius R_t and center c_t of the ball at each iteration, while this is considerably faster than solving the QP of conventional SVM training, it stills requires solving a quadratic subproblem defined on the core-set (which for some datasets can be large). To cope with this problem, a simplified algorithm was proposed in [83] where the radius of the MEB is fixed so at each iteration only the center of the ball is recomputed.

In order to evaluate the performance of Core Vector Machines applied to speech processing, a comparison will be carried out between CVMs and state-of-the-art SVM training for cepstral features classification.

7.1.2 Experimental setup

Core Vector Machines are especially suited to deal with large datasets and it is in this context where they have shown considerable advantages respect to the use of conventional SVMs. For this reason, Switchboard [84] that comprises tens of hours of speech, has been selected as the training material[31] while CallHome, another corpora comprised of telephone speech, has been selected for evaluation purposes. .

The experiment consists of training a pairwise classifier to separate cepstral feature vectors aligned to the central state of the phone AA from cesptral feature vectors aligned to the central state of the phone HH. For this reason, Sonic [41] has been used to perform a forced aligned to the data and obtained the state-alignment information. The phonetic classes AA and HH appear with a similar frequency in the training material so produce a balanced training set. In addition, based on the experiments done for children's speech, these classes are expected to be relatively easy separable thus allowing the training of traditional SVMs in a reasonable time span. An initial step consisting of a Viterbi alignment between the audio and the transcriptions has been carried out using Sonic [41]. The total number of training samples aligned to each class is 399761 and 503639 respectively. The resulting number of test samples available for both classes is 43268.

Using these data points, several diluted training sets have been created corresponding to several diluting factors $\mathbf{1} : \mathbf{x}$. A diluting factor $\mathbf{1} : \mathbf{x}$ means that 1 sample is selected out of $\mathbf{X}$ consecutive samples in the original training set. This is a natural mechanism to obtain datasets that contain only a percentage of the samples in the original dataset that can be used to evaluate how the training complexity evolves as the number of training samples increases. For each diluted set three classifiers have been trained:

- A state-of-the-art SVM classifier using the LibSVM library [58].
- A CVM classifier for which the MEB's radio is recalculated on each iteration as described in [82] . The library used is LibCVM [85].
- A CVM classifier for which the MEB's radio is given in advance as described in [83], this

[31] Both the predefined training and test partitions of the Switchboard corpora have been combined and used as training in order to have a dataset as big as possible

classifier is referred as BVM. The library used is LibCVM [85].

These classifiers have been trained using a Gaussian Radial Basis Function. The parameters C and γ have been selected doing a 5-fold cross validation process. The resulting optimal values are $C = 1$ and $\gamma = 0.125$, which have been used to train all the classifiers. For the conventional SVM classifiers a tolerance of termination criterion $\epsilon = 10^{-3}$ has been used. In the case of CVM and BVM, values of ϵ up to 10^{-4} have been reported [83] to produce the best results for a wide variety of datasets. However, it has been found that such a large value of ϵ does not produce satisfactory results for cepstral features classification, so a smaller value of $\epsilon = 10^{-5}$ have been used at the expense of increasing the training time.

Experiments have been carried out using a machine running the Windows XP operative system with 1GB of main memory and 3.00 GHz. For all the experiments it has been made sure that all the computations were conducted in main memory so no swapping was allowed. The classifiers trained have been compared in terms of training time, number of support vectors generated and classification accuracy.

7.1.3 Results

Table 7.1.3 and, graphically, figure 7.1 show the training time for the three different training techniques and seven different diluting factors. As can be observed, for a number of training samples up to 180680, disregarding the technique used, the training time grows nonlinearly with the number of samples. However, once the number of samples goes beyond that value, the training time of CVM and BVM classifiers becomes linear with the number of samples while the training time of the SVM classifier remains non linear. So it is possible to see an asymptotic linear behavior that is especially convenient for dealing with large datasets.

An analogous behavior is observed when analyzing the number of support vectors (see table 7.1.3 and figure 7.2). In this case, for a number of samples up to 180680, the number of support vectors grows linearly with the number of samples independently of the technique employed. However, when training on the whole set of samples, the conventional SVM classifier still introduces a considerable number of new support vectors, while both CVM and BVM only add a relatively small number of them.

Diluting factor	#samples	SVM	CVM	BVM
1000	903	0.079	0.16	0.34
500	1807	0.25	0.5	0.5
100	9034	6.7	6.95	5.36
50	18068	22.94	24.61	17.38
10	90340	504.41	2243	1184.05
5	180680	1888.38	8629	2443.06
1	903400	67342	10756	3665.38

Table 7.1: *Training time for different training techniques.*

The explanation is that, when the number of training samples is not big enough (below 200.000 in the experiments conducted), new training samples still bring in additional information useful for classification, which makes the number of core vectors and thus the number of support vectors and the training time grow as the number of training samples grow. However, once the number of training samples used reaches some point (that it is believed is task dependent), if more samples are added, only a very small percentage of them contribute to improve the classification, so only a relatively small number of extra support vectors are needed. While both CVM and BVM are able to exploit this situation in terms of training time and number of support vectors produced (note that these concepts are strongly connected) by limiting the number of core-vectors contained in the core-set, conventional SVMs training does not, thus producing a huge number of support vectors that do not make a difference in the classification accuracy. This point is reflected in table 7.1.3, in which for the larger datasets, CVM and BVM techniques produce comparable classification accuracy to conventional SVMs while considerably reducing the number of support vectors.

Based on these experiments for cepstral features classification, it can be concluded that, although for small and middle size datasets the MEBs-based techniques do not show any advantage respect to conventional SVM training (and in some cases may slightly deteriorate the performance), their asymptotic linear behavior clearly makes them a better choice for training on very large datasets. It would have been interesting to train on even larger datasets, however the main memory limitations of the machine used prevented that possibility. Additionally, the training time of conventional SVMs on the largest datasets would have made the process almost unfeasible.

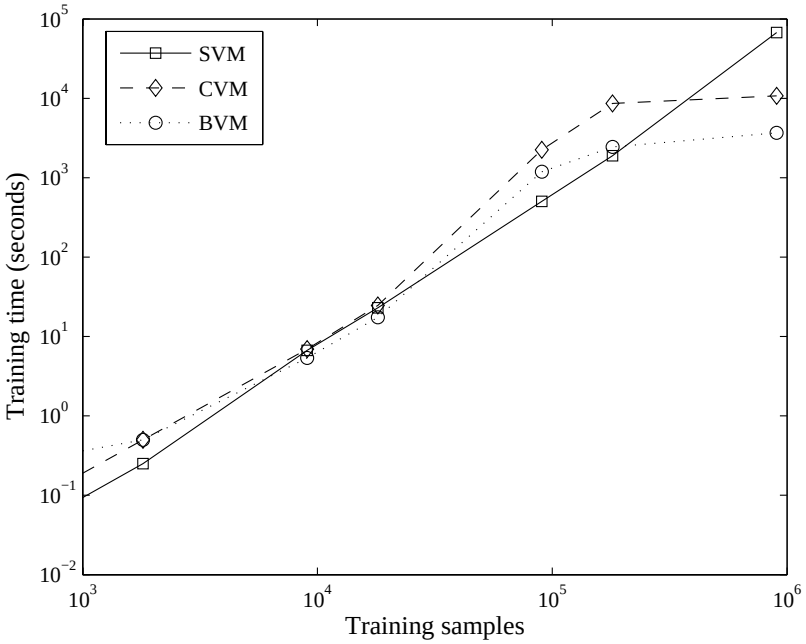

Figure 7.1: *Training time as a function of the number of training samples (data corresponds to table 7.1.3)*

Diluting factor	#samples	SVM	CVM	BVM
1000	903	269	315	352
500	1807	482	621	657
100	9034	1367	2095	1773
50	18068	2320	3505	2844
10	90340	9494	10756	8323
5	180680	17486	14645	10805
1	903400	77293	15372	11962

Table 7.2: *Resulting number of support vectors for different training techniques.*

Diluting factor	#samples	SVM	CVM	BVM
1000	903	88.47	87.6	87.9
500	1807	89.44	88.24	88.37
100	9034	90.37	90.28	90.21
50	18068	90.64	91.07	90.47
10	90340	91.56	91.26	91.8
5	180680	91.84	92.05	92.61
1	903400	92.55	92.68	93.43

Table 7.3: *Classification accuracy for different training techniques.*

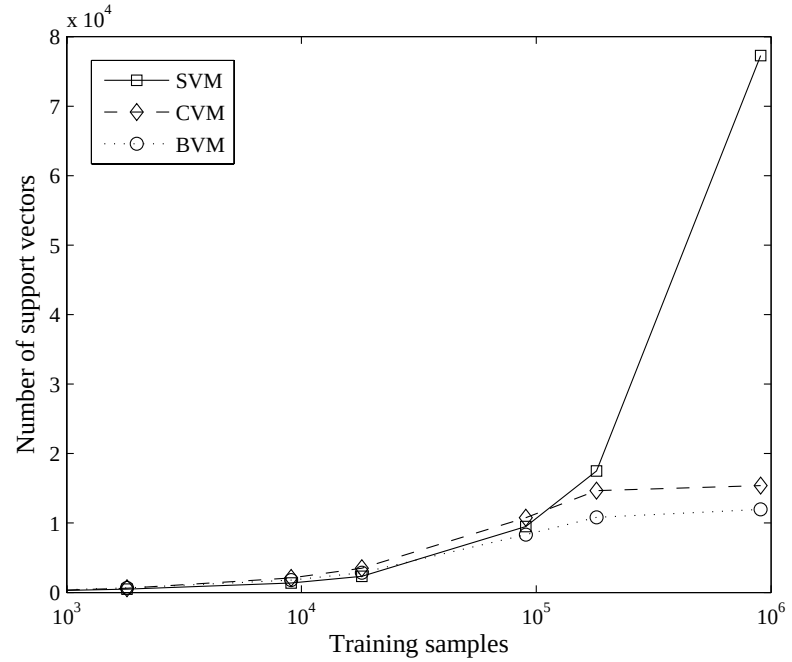

Figure 7.2: *Number of support vectors produced as a function of the number of training samples.*

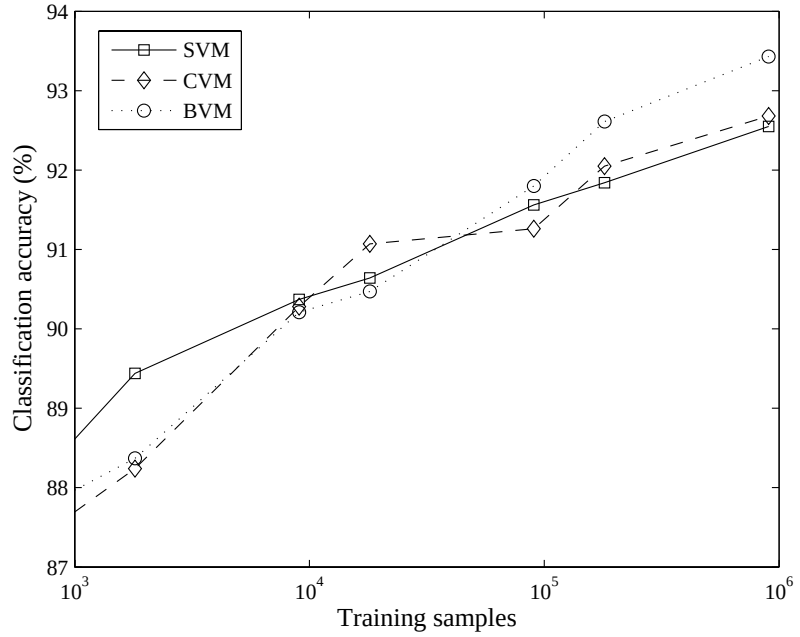

Figure 7.3: *Classification accuracy as a function of the number of training samples.*

7.1.4 Working with a preset number of support vectors

One interesting point of CVMs is that they allow to set in advance the maximum number of support vectors the training produces by limiting the maximum number of core-vectors in the core-set. Once the training is done, the resulting number of support vectors is a lower bound of the preset number. This is a very interesting feature since, the evaluation of the decision function of a SVM (which strongly influences the decoding time of the SVM/HMM system proposed) can be considered as linear with the number of support vectors. However, it is unclear up to what extent can the maximum number of support vectors be limited without compromising the classification accuracy. In order to shed some light on this issue, some informal experiments have been carried out, unfortunately it has been observed that no significant reduction in the number of SVs is possible without deteriorating the classification accuracy. Nonetheless, an in depth analysis and a more detailed set of experiments should be conducted to draw conclusion on this matter.

7.2 Decoding time scalability

In a conventional state-of-the-art speech recognition system, decoding time is mainly determined by two factors:

- Viterbi beam-search: time necessary to build the search space and expand/prune it.
- Computation of the emission probabilities.

During a conventional Viterbi search, an emission probability needs to be computed for each of the HMM-states that remain active at the current time frame. While the number of active HMM-states depends on several factors like the size of the lexicon and the beam pruning, usually, the emission probability of most of the HMM-states needs to be computed. In the case of a SVM/HMM system with emission probabilities calculated using 4.1 and 4.5, this results in the evaluation of most the pairwise SVM classifiers at any give time frame, which strongly deteriorates the real time performance. For example, to estimate 4.1 using 52 phonetic classes and three-state HMMs, if an average of 80% of the states are active, 11625 SVM classifiers need to be evaluated at any given time frame. This represents 96% of the total pairwise models trained and is computationally extremely

expensive. It is believed that this is one of the most serious shortcomings in the applicability of SVMs for speech decoding following this probabilistic scheme.

Given that the core of the problem is the time required to evaluate such a big number of SVM classifiers during the computation of the emission probabilities, improving the real time performance of the decoder can be carried out in two complementary ways:

- Reducing the evaluation time of the individual SVM classifiers.
- Reducing the number of classifiers that need to be evaluated at the frame level.

This section is devoted to explore solutions from both perspectives in order to improve the decoding time scalability.

7.2.1 Reducing the evaluation time of the individual SVM classifiers.

As introduced in 2.3.3, the time-complexity of evaluating a SVM classifier for a given input sample $\mathbf{x}$ is dominated by the number of kernel evaluations. Particularly, the time-complexity is linear with the number of kernel evaluations required to classify the input sample. While in the case of binary classification the number of kernel evaluations corresponds to the number of support vectors of the classifier, in a multiclass classification problem for which several binary classifiers contribute to the final call, the number of kernel evaluations is proportional to the number of unique support vectors across classifiers. As introduced in 2.3.3, the justification is that kernel evaluations $K(\mathbf{s}_i, \mathbf{x})$ between the input sample $\mathbf{x}$ and multiple occurrences of the same support vector $\mathbf{s}_i$, can be cached and reused.

In the case of a SVM/HMM speech recognition system, the number of different classes (recall the correspondence between classes and HMM-states) that intervene in the multiclass classification is very high. Assuming that the one-versus-one strategy has been selected, the number of pairwise classifiers trained for each class is consequently high as well. Thus, the number of times a support vector may appear in different classifiers is potentially high, and so it is the number of kernel predictions that can be reused.

Now, this circumstance will be explored for the experimental setup described in 4.3.1. Table 7.4 shows a comparison between the number of support vectors and the corresponding unique

support vectors for two HMM architectures. As can be seen, in the case of a one-state HMM architecture, the number of support vectors (total number of support vectors across the 1326 classifiers trained) is 11.16 times higher that the total number of unique support vectors. Thus, an impressive overall reduction in the time required for computing the emission probabilities of 91% can be potencially attained by reusing kernel evaluations at the frame level. In the case of a three-state HMM architecture an even higher reduction of about 96% can be obtained motivated by the higher number of classifiers trained for each class.

Parameter	one-state HMMs	three-state HMMs
Number of phonetic classes	52	52
Number of classes	52	156
Number of classifiers trained	1326	12090
Number of classifiers per class	51	155
Number of support vectors	8912183	21567545
Number of unique support vectors	798787	850565
Duplication ratio	11.16	25.37

Table 7.4: *Comparison of the number of support vectors and unique support vectors for two HMM architectures.*

However, in practice, it is not possible to reach these impressive reduction factors when computing the emission probabilities. The reasons are the following:

First, besides the kernel evaluations, there are other less costly operations that still need to be carried out to evaluate the decision function linked to a SVM. To illustrate this point, equation 7.2 shows the SVM decision function, which sign is used to classify an input sample $\mathbf{x}$. Recall from section 2.1 that N_S is the total number of support vectors, s_i are the support vectors and α_i are the Lagrange multipliers.

$$f(\mathbf{x}) = \sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b \tag{7.2}$$

As can be seen, even reusing a big percentage of the kernel predictions across classifiers, all the products with the Lagrange multipliers still need to be carried out to evaluate the decision function of each classifier.

Second, the computation of emission probabilities implies a set of operations on top of the decision function evaluations:

- In order to obtain a class-posterior probability $p(k_i|k_i \, or \, k_j, \mathbf{x})$ from each pairwise classifier, it is necessary to map the distance obtained from the SVM to a probabilistic value using 2.20.
- Pairwise probabilities $p(k_i|k_i \, or \, k_j, \mathbf{x})$ obtained from each pairwise classifier $SVM_{(k_i,k_j)}$ need to be combined to obtain the final class posteriors $p(k_i|\mathbf{x})$ using 4.1 and then, the emission probabilities $p(\mathbf{x}|k_i)$ using 4.5.

Finally, it is important to consider that if two classifiers share a support vector $\mathbf{s}_i$ but are trained with different kernel parameters (for example, a different value of γ in the case of the Gaussian kernel), kernel predictions $K(\mathbf{s}_i, \mathbf{x})$ for the support vector $\mathbf{s}_i$ and an input sample $\mathbf{x}$ can not be directly reused. In this case, however, a simple trick allows the reuse of kernel predictions by introducing only an extra operation. Let be $SVM_{(k_i,k_j)}$ a classifier trained to separate classes k_i and k_j that was trained using a Gaussian Radial Basis Function $K_1(\mathbf{x}, \mathbf{y}) = e^{-\gamma_1 \|\mathbf{x}-\mathbf{y}\|^2}$. Let be $SVM_{(k_i,k_l)}$ a classifier trained to separate classes k_i and k_l that was trained using a Gaussian Radial Basis Function $K_2(\mathbf{x}, \mathbf{y}) = e^{-\gamma_2 \|\mathbf{x}-\mathbf{y}\|^2}$. Let be $\mathbf{s}$ a training sample from class k_i that is support vector in both classifiers, and $\mathbf{x}$ an input sample to classify. In this case $K_2(\mathbf{s}, \mathbf{x})$ can be easily obtained from $K_1(\mathbf{s}, \mathbf{x})$ as expressed in 7.3.

$$K_2(\mathbf{s}, \mathbf{x}) = e^{-\gamma_2 \|\mathbf{s}-\mathbf{x}\|^2} = \left(e^{-\gamma_2 \|\mathbf{s}-\mathbf{x}\|^2}\right)^{\frac{\gamma_1}{\gamma_1}} = \left(e^{-\gamma_1 \|\mathbf{s}-\mathbf{x}\|^2}\right)^{\frac{\gamma_2}{\gamma_1}} = K_1(\mathbf{s}, \mathbf{x})^{\frac{\gamma_2}{\gamma_1}} \tag{7.3}$$

As can be seen, only an additional exponentiation is needed to reuse kernel predictions across classifiers that make use of a different γ parameter in the Gaussian kernel.

7.2.2 Reducing the number of classifiers that need to be evaluated at the frame level.

In this section it will be described an algorithmic method to dynamically select the order in which the pairwise classifiers are used in the computation of the emission probabilities so that at any given time frame only a reduced subset of the total classifiers trained is used. The idea is to perform the computation globally using a greedy strategy in which the pairwise classifiers that are expected to be the most discriminative are evaluated first. This procedure is expected to rapidly assign a very low probability to unlikely HMM-states by evaluating first the pairwise classifiers of the best

scoring HMM-states. During the process, once the partial emission probability computation of an HMM-state drops below a certain value (relative to the probability of the best scoring HMM-state), no more pairwise classifiers need to be evaluated for that state and it can be pruned from the computation process at an early stage.

1. Create a list L_{active} with pairs (k_i, μ_i) where k_i represents each of the classes (i.e. HMM-states of the different phone classes) that remain active in the Viterbi search at the current time frame, and μ_i (initialized to 0) will be used to keep a partial summation of pairwise probabilities for the class k_i.

2. Create an empty list L_{final} to keep the classes that will be used do the final computation of probabilities. Another empty list L_{pruned} is created to store the classes that are pruned from the global computation of probabilities.

3. Initialize (k_b, μ_b) with the pair (k_i, μ_i) with smallest value of μ_i. In the first iteration (i.e. when all the μ_i are 0) k_b is initialized with the best scoring class from the previous time frame (this is the class that has the highest expected probability value for the current time frame). k_b will be used to keep the class with the best partial probability estimation.

4. For each pair $(k_i, \mu_i)_{i \neq b}$ in L_{active} do the following:

   (a) compute $p(k_b | k_b \, or \, k_i, \mathbf{x})$ using the corresponding pairwise classifier. If no more pairwise classifiers are left to evaluate for k_b, move (k_b, μ_b) to L_{final} and go to step 5.

   (b) $\mu_i = \mu_i + \frac{1}{1 - p(k_b | k_b \, or \, k_i, \mathbf{x})} - 1$

   (c) $\mu_b = \mu_b + \frac{1}{p(k_b | k_b \, or \, k_i, \mathbf{x})} - 1$

   (d) if $p(k_b | k_b \, or \, k_i, \mathbf{x}) < 0.5$ then $(k_b, \mu_b) = (k_i, \mu_i)$

5. If L_{active} contains less than two elements move them to L_{final} and go to step 8.

6. Sort the elements of L_{active} in ascending order of μ_i. Note that μ_i represents part of the overall summation of pairwise probabilities that is done in the right term of expression 4.1 for calculating the posterior of k_i. Thus, $\mu_i + 1$ can be considered as the inverse of a partial estimation of the posterior of k_i.

7. For each element k_i in L_{active} if $\frac{\mu_i+1}{\mu_b+1} > \lambda$ move the element to L_{pruned}. Go to step 3.

8. For each element (k_i, μ_i) in L_{pruned} approximate the posterior probability $p(k_i|\mathbf{x})$ of k_i as a minimum value relative to $\frac{1}{\mu_b+1}$ (that represents the best partial probability estimation). This is expressed in (5).

$$p(k_i|\mathbf{x}) = \frac{1}{(\mu_b + 1)\lambda} \tag{7.4}$$

9. For each element (k_i, μ_i) in L_{final} compute the posterior probability $p(k_i|\mathbf{x})$ of k_i using only the pairwise classifiers of the classes contained in L_{final}. This is expressed in (6), where F represents the number of elements in L_{final}.

$$p(k_i|\mathbf{x}) = \left[\sum_{\substack{j \neq i, \\ j \in L_{final}}} \frac{1}{p(k_i|k_j \; or \; k_i, \mathbf{x})} - (F-2) \right]^{-1} \tag{7.5}$$

10. According to 4.5 a final step of normalizing the posteriors by the phone priors is necessary.

Note that using a high enough value of λ this algorithm is completely equivalent to the conventional calculation using 4.1 and 4.5 for every active HMM-state. Another interesting point is that, as described in the step 9 of the algorithm, the posterior probabilities of the "surviving candidates" ($k_i \in L_{active}$) are computed using only pairwise classifiers of the classes contained in L_{final}. There are two reasons for that:

- It allows a more homogeneous computation of probabilities since at step 9 the number of pairwise classifiers already evaluated for classes in L_{final} vary from one class to another.
- Probabilistic values from pairwise classifiers that separate more likely classes (those contained in L_{final}) are expected to be more reliable.

Experiments

Experiments have been carried out to evaluate the effectiveness of the algorithm previously proposed to reduce the number of pairwise classifiers that need to be estimated at each time frame during

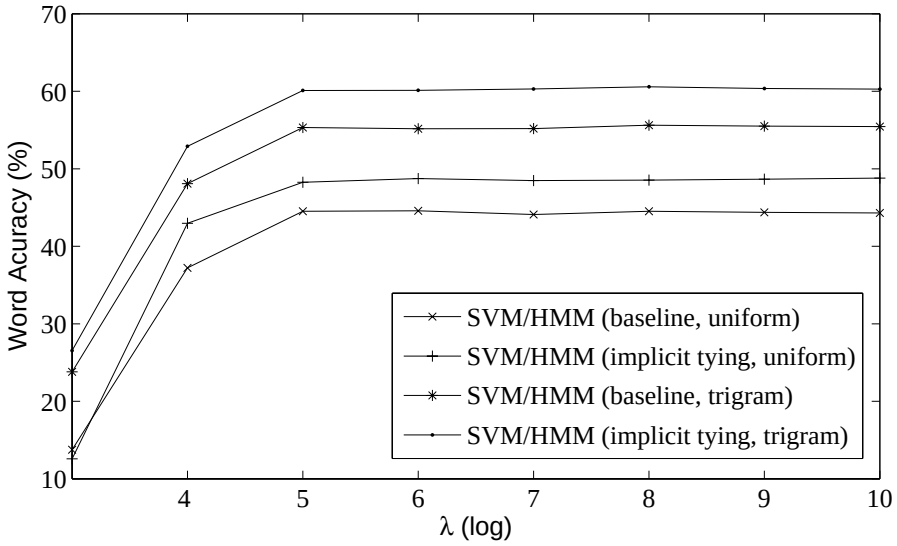

Figure 7.4: *Word accuracy for different values of* λ

decoding. The SVM/HMM system used here is the one described in section 4.3.1, making use of state-tying as described in chapter 5. In this system, the emission probabilities of each HMM-state are globally calculated according to the agorithm previously described. Different decoding processes have been carried out for which different values of λ have been used. As can be seen in Fig. 7.4 the value of λ can be set at 10^5 with no loss in recognition accuracy. In particular, for such a value of λ only an average of 14% of the pairwise models that, according to expressions 4.1 and 4.5 would be evaluated, are used at each time frame. However, significantly higher values of λ still allow a large reduction in the number of pairwise classifiers. For example if λ is set to 10^8 only an average of 38% of the pairwise models are used.

From these results it is possible to conclude that the proposed algorithm dramatically reduces the number of pairwise classifiers that need to be evaluated at the frame level while preserving the recognition accuracy.

Similarities with the DDAGs method

It is interesting to mention that this algorithm presents some similarities respect to the DDAGs method for multiclass classification [86]. In fact, it can be seen that there is an implicit graph structure that is built as the pairwise classifiers are evaluated. The DDAGs method can be considered

as a particular case of the algorithm proposed in which the computation of the probability $p(k_i|\mathbf{x})$ of a class k_i is stopped once it falls below 0.5.

The main differences between the proposed method and DDAGs are listed below:

- Pairwise classifiers of a given class k_i are evaluated until the probability of that class falls below a given threshold, which is relative to the most likely class in the partial computation. In the DDAGs method, only a pairwise classifier of each class is evaluated, with the exception of the winning class for which all the pairwise classifiers are evaluated. It is for this reason that DDAGs are not suitable for computing class-level posteriors other than that of the winning class.

- In opposition to the DDAGs method in which the graph is built in advance, at each iteration there are multiple pairwise classifiers that can be evaluated (those of all the active classes k_i in the list L_{active}, for this reason a criterion needs to be established to select the next pairwise classifier to evaluate. In the proposed algorithm the criterion consists of evaluating first pairwise classifiers of the most likely class in the partial computation.

Finally, note that while this algorithm has been proposed for emission probabilities computation it can be directly applied to any multiclass classification problem in which class-posteriors are required and SVM pairwise classifiers are selected as the classification strategy.

7.3 Models' size

Whenever a system that performs a large number of computations is expected to work in real time, and that is the case of a speech decoder, it is of foremost importance that the core of the system is able to work in main memory. In the case of a decoding system, three main factors govern the memory requirements:

- Acoustic models' size: a conventional GMM/HMM system requires the storage in memory of roughly 2000 triphone models, while the size of these models depends mainly on the model topology, the number of mixtures used and the strategy selected for tying parameters across triphones, their memory requirements are nowadays not a big issue. However, in the case

of a SVM/HMM decoder like the one introduced in section X in which the one-versus-one strategy is selected for multiclass classification, a very large number of SVM classifiers need to be trained and be present in main memory at any given time frame during decoding. Given that the size of a SVM classifier is proportional to the number of support vectors and this number is, as will be shown later on, relatively large, a compression mechanism is needed to limit the main memory necessary to store the classifiers.

- Language models' size: depending on the application, and especially if a n-gram of third or superior order is applied, a large vocabulary continuous speech recognizer (LVCSR) may need considerable space to store the probabilities of all the possible n-grams.

- Search space size: the search space size is typically determined by the size of the lexicon, the probabilistic language model utilized and the pruning strategy and beam width selected. State-of-the-art GMM/HMM decoders apply different techniques during the Viterbi search, like language model or phoneme look-ahead, aiming for a reduction of the search space thus accelerating the decoding process. Analogous techniques can be successfully incorporated into the proposed SVM/GMM system so in this respect no significant differences have been observed between both systems in term of memory usage.

It's clear that it is the first of these factors the one that requires a careful study in the proposed system.

7.3.1 Compression method

It has been observed in previous sections that the total number of support vectors resulting from the training of all the one-versus-one classifiers for a given class (HMM-state) is considerably higher than the total number of training samples available for that class. Given that a support vector is by definition a training sample, it is clear that many training samples become support vectors of different classifiers so there is a high degree of redundancy of support vectors across pairwise classifiers trained for the same class.

A compression method has been proposed that consists of extracting the support vectors contained inside each SVM classifier, unifying duplicated support vectors, and storing them alto-

gether in a shared repository. Following this procedure, each support vector resulting from the training of a classifier is replaced by a pointer to the exact same support vector in the repository. The benefit of this compression method obviously lies in the much smaller space needed to store a pointer (an index in disk) than a support vector (39 floating point coefficients). Since most of the models have a great number of support vectors in common, a big decrease in the storage requirements (both in disk and in main memory) is expected by storing each support vector only once. Following this approach, the maximum number of support vectors that need to be stored for a given class will be at most the number of available samples for that class. However, as it will be shown next, the total number of support vectors that need to be stored for each class is only a percentage of the available ones.

Parameter	Original models	Compressed models
Number of phonetic classes	52	52
Number of models	1326	1326
Average number of samples per class	17498	17498
Average number of support vectors per class	171388	15361
Average model's size	2.05MB	78.8KB
Total size	2720MB	102MB + 118.8MB

Table 7.5: *Comparison of storage requirements between the original SVM models and those resulting from the compression algorithm proposed.*

7.3.2 Conclusions

In this chapter several scalability and real time performance issues have been discussed. In that respect, the main weaknesses of the SVM/HMM speech recognition system proposed have been identified and different techniques have been proposed to alleviate them. The following list summarizes the topics discussed:

- Training time scalability: Core Vector Machines [82] and Ball Vector Machines [83] have been explored as approximate SVM solvers that exhibit training time asymptotically linear with the number of samples. Experiments carried out show that these techniques present an excellent performance for feature frames classification in very large datasets. They show comparable classification accuracy than standard SVM-training techniques while producing

considerable fewer support vectors. Hence, they are good candidates for training SVM-based acoustic models in large datasets.

- Decoding time scalability: the decision function of a SVM, and thus the SVM-acoustic models proposed in this thesis work, is expressed as a function of a subset of the training samples called support vectors. As the training material increases the number of support vectors tend to increase what strongly deteriorates the real time performance. This problem has been successfully addressed from two different perspectives:

  1. Reusing kernel predictions across classifier evaluations.
  2. Reducing the number of classifiers that need to be evaluated at the frame level using an heuristic approach. This heuristic approach not only reduces dramatically the number of classifiers that need to be evaluated but can be potentially to any multi-class classification problem that requires probability estimates for the most likely classes.

- Models' size: it's been taken advantage of the redundancy of support vectors across classifiers to store them in a unique repository that can be accessed through pointers. This procedure limits the models' size to the total number of unique support vectors, which strongly reduce the storage requirements.

Chapter 8

Summary, Conclusions and Future Work

This thesis is focused on exploring the applicability of Support Vector Machines to the task of continuous ASR. While there exist in the literature a number of successful applications of SVMs to different speech processing tasks, their application to continuous speech recognition is still a quite unexplored issue.

8.1 Summary of results

This thesis work starts making an introduction to the mathematical foundations of SVMs and to different strategies proposed in the literature for dealing with probabilities estimation and multiclass classification. These issues are particularly interesting from the point of view of speech recognition, in which the number of classes (HMM-states) is quite large and probabilistic values, and not just class labels, are much needed. Different techniques are discussed and compared from several angles. Finally, the one-versus-one strategy has been selected for dealing with multiclass classification. This technique presents both superior accuracy and trainability advantages respect to competing techniques. In the other hand, while this technique results in a very large number of classifiers (especially if the number of classes is large, and that is the case of speech recognition), it allows a dynamic selection of pairwise classifiers during decoding. Chapters 5 and 7 show, respectively,

that an appropriate dynamic selection of pairwise classifiers leads to a better recognition accuracy and a dramatic reduction in the decoding time.

In chapter 3, a review on the state-of-the-art of SVM-based speech processing techniques is carried out, in which the main challenges in the application of SVMs to continuous speech recognition are outlined. Also in that chapter, a novel technique for pronunciation scoring using SVMs as probabilistic estimators and phone-graphs is introduced. The proposed technique significantly outperforms state-of-the-art techniques, hence showing the potential of SVMs as probabilistic estimators.

Chapter 4 is dedicated to introduce a novel SVM/HMM stand-alone speech recognition system and to evaluate it experimentally. This system has shown superior recognition accuracy than a comparable GMM/HMM system. In the other hand, the proposed system makes use of a much bigger number of parameters, which compromises its real-time performance and scalability. In this respect, chapter 7 is dedicated to, first, identify the system's scalability problems and, second, address them from three different perspectives: training time, decoding time and models' memory requirements.

8.2 Contributions

This section summarizes the main contributions of this thesis work.

- A stand-alone SVM/HMM continuous speech recognition system has been proposed and fully implemented showing superior word accuracy to a conventional GMM/HMM in the experimental framework under consideration. The proposed system, published in [87], presents the following properties:
  - It does work as a stand-alone system. In fact, its only connection with a GMM/HMM system is the state-alignment required in the segmentation of the training data used to train the SVM classifiers. However this connection is not that because hand-labeled speech corpus like TIMIT can potentially be used to generate a set of bootstrap models.
  - It performs speech recognition and not hypothesis rescoring (in contrast with other systems like [32]).

- It is a continuous speech recognition system (in contrast with isolated recognition approaches like [34] or [35]).
- It does not make use of transition probabilities extracted from acoustic models trained using a conventional GMM/HMM system (in contrast with the system proposed in [36]). For this reason, it can not be considered a hybrid approach but an independent system.

While the SVM/HMM system proposed shows superior word-recognition performance than a conventional GMM/HMM system it presents some disadvantages in terms of real time performance and scalability (both motivated by the large number of parameters). The proposed system is not intended to replace conventional GMM-based speech recognition systems but to take a step forward toward the application of SVMs to continuous speech recognition.

- A procedure for implicit state-tying of pairwise classifiers during decoding has been proposed with the intention of removing unreliable pairwise classifiers from the computation of emission probabilities. This procedure presents the following advantages:
  - Word accuracy improvement: The tying procedure helps to reduce the WER, indicating that training classifiers to separate very similar classes is not recommended.
  - Faster decoding: fewer classifiers need to be evaluated at the frame level during the Viterbi search, with the consequent reduction in computation.
  - Faster training: after doing the cross-validation process those classifiers with an accuracy below the threshold do not need to be trained.
- It has been carried out a study on the integration of context-dependency modeling into the proposed SVM/HMM decoder. A SVM-based triphone clustering method based on the alternative expansion of two binary trees using phonetic rules has been described and experimentally evaluated. The proposed method is able to effectively identify clusters of triphones which pairwise separability is clearly superior than that of the corresponding context independent classifier. However its feasibility for continuous speech decoding has not been evaluated and it has been left for future work.

- It has been presented an algorithm that allows the computation of posterior probabilities for each of the classes involved in a multiclass classifications task. This technique has been experimentally evaluated in the task of emission probabilities computation showing that the probabilities produced are as accurate as those computed using state-of-the-art alternative techniques while dramatically reducing the number of pairwise classifiers that need to be evaluated. The algorithm proposed, although evaluated in the case of emission probabilities computation, can be potentially applied to any multiclass classification problem involving a large number of classes and pairwise classifiers.

- Several scalability issues have been identified in the SVM/HMM recognition system. These issues have been addressed from different angles:

  - Training time scalability: two approximate SVM solvers [82] [83] have been explored for feature frames classification on large datasets. They both show comparable classification accuracy than standard SVM-training techniques while producing considerable fewer support vectors. Hence, they are good candidates for training SVM-based acoustic models in large datasets, which represents one of the main concerns in the application of SVMs to speech processing tasks.
  - Decoding time scalability: this problem has been successfully addressed from two different perspectives:
    1. Reusing kernel predictions across classifier evaluations.
    2. Reducing the number of classifiers that need to be evaluated at the frame level using an heuristic approach.
  - Models' size: a repository is utilized to uniquely store support vectors shared across different classifiers.

  Note that these scalability issues are not specific to the proposed SVM/HMM speech decoding system but to a much wider range of speech applications that make use of feature frames classification under the pairwise-coupling framework.

- It has been presented a successful application of SVMs to pronunciation scoring in the context

of a children's word reading task. The idea consists in extracting phone-level and frame-level SVM-based posterior probabilities from a phone graph and combine them to obtain pronunciation scores. The proposed technique and features shows a 22.9% relative error reduction respect to conventional log-likelihood based techniques. This application has been published in [88].

8.3 Future work

- Reducing the number of parameters: it is believed that the main shortcoming of the proposed SVM/HMM system is the number of parameters. SVMs are a non parametric modeling technique which solution (decision function) is expressed using a subset of the training samples called support vectors. For complex classification problems, like speech features classification, typically a complex decision function and thus a great number of support vectors are required to attain satisfactory classification results. The one-versus-one technique has been selected across this thesis work as the strategy for dealing with multiclass classification. While the application of this technique results in small classifiers that can be trained faster and are as accurate as those resulting from the one-versus-rest strategy, the total number of resulting classifiers is in the order of several thousands. Several techniques have been proposed in this thesis work to deal with this issue when computing the emission probabilities, like caching kernel evaluations and dynamic selection of classifiers. As shown in chapter 7.2, these technique have shown to dramatically reduce the cost of computing the emission probabilities. Nevertheless, it would be interesting how the real time performance scales as the size of the training material increases (for example, in larger corpora of tens of hours of speech or more).

- Context-dependency: a method for context-dependent acoustic modeling using SVMs has been proposed in the chapter 6 of this thesis. While this method has been partially evaluated and, thus, its feasibility has been partially proven. A complete evaluation of that method (i.e. recognition accuracy compared to the equivalent context-independent SVM/HMM system and/or to a context-dependent GMM/HMM system) has not been carried out because it would require too much effort by itself to be included as part of this thesis work. Doing such

work is left for future research.

- Mismatched conditions: it is well known that the accuracy of speech recognition systems rapidly degrades when deployed in acoustical environments different than those used in training [89][90]. For example using a speech recognition system from the cellphone introduces a great acoustic variability because the noise in the network itself, the background noise, etc. In these cases, the conditions in which the system is used may enormously differ from in which the acoustic models are trained, which may make the system unusable. While there exist a number of technique to deal with the problem of mismatched conditions, which can be broadly divided into two main categories, feature and model adaptation, there are situation in which the environment conditions vary very rapidly and, thus, those adaptation techniques have a limited effect. In these cases the robustness of the original model and its ability to generalize is of major importance. In this respect, one of the most appealing properties of SVMs is their generalization capability, it is for this reason that SVM-based acoustic modeling seems very promising for dealing with mismatched conditions.

- Comparison with a conventional GMM/HMM system which acoustic models are trained using a discriminative criterion: as it was mentioned in chapter 4, the utilization of SVMs for acoustic modeling in a SVM/HMM system shares some of the motivations of using discriminative training criteria for the training of GMM/HMM acoustic models. For this reason, it would be interesting to carry out a comparison between such systems and not restrict the comparison to a system trained under Maximum Likelihood. This is left for future work.

- Conditional Random Fields: conditional random fields (CRF) are discriminative models that, unlike HMMs, do not attempt to model the probability distribution of the input data (the acoustic observations) but the probability of sequences of labels. Conditional Random Fields allow the computation of the posterior probability $P(Y|X)$ of a sequence of class labels Y given a sequence of input samples X and their attributes. For this reason, CRFs seem to be a more suitable mechanism to integrate frame-level posteriors (which are the natural output of SVMs) into the acoustic modeling than HMMs. Recall from section 4.2.1 that SVM-based frame-level posteriors $p(y|x)$ are transformed to class conditional probabilities dividing by

class priors. Although this mechanism has been previously used in the literature in the case of hybrid ANN/HMM systems [70], it seems less convenient. For example, in [99] CRFs are used for integrating local discriminative classifiers.

8.4 Main publications

The following list contains the most relevant publications made during this thesis work:

- D. Bolanos and W. H. Ward, "Implicit State-Tying for Support Vector Machines Based Speech Recognition". *In Proceedings of Interspeech 2008*, September 22-26. Brisbane, Australia.

- D. Bolanos, W.H. Ward, B. Wise and S. Van Vuuren, "Pronunciation Error Detection Techniques for Childrens Speech". *In Proceedings of Interspeech 2008*, September 22-26. Brisbane, Australia.

- D. Bolanos, W.H. Ward, Sarel Van Vuuren and Javier Garrido, "Syllable lattices as a basis for a Children's Speech Reading Tracker", *In Proceedings of Interspeech 2007*, August 27-31. Antwerp, Belgium.

- D. Bolanos and W.H. Ward, "Posterior Probability Based Confidence Measures Applied to a Children's Speech Reading Tracking System", *In Proceedings of NODALIDA 2007, the 16th Nordic Conference of Computational Linguistics*. May 25-26 2007. Tartu, Estonia.

- D. Bolanos and W. H. Ward, "SVM-based Posterior Probabilities for Syllable Confidence Annotation", *In proceedings of V Jornadas en Tecnologa del Habla*. November 12-14 2008. Bilbao, Spain.

Bibliography

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

[2] M.A. Aizerman, E.M. Braverman and L.I. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning". Automation and Remote Control, 25:821837, 1964.

[3] J. Yousafzai, Z. Cvetkovic, P. Sollich and M. Ager, "Robustness of phoneme classification using support vector machines: a comparison between PLP and acoustic waveform representations". In Proceedings of ICASSP 2008.

[4] C.J. Burges, 1998. A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. 2 (2), 121167.

[5] J. Platt,"Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". In Advances in Large Margin Classifiers. Cambridge, MA: MIT Press, 1999.

[6] A. Madevska-Bogdanova, D. Nikolikb and L. Curfsc, "Probabilistic SVM outputs for pattern recognition using analytical geometry". Neurocomputing. v62. 293-303.

[7] J. Weston and C. Watkins, "Multi-class support vector machines". Presented at the Proc. ESANN99, M. Verleysen, Ed., Brussels, Belgium, 1999.

[8] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems". Comput. Learing Theory, pp. 3546, 2000.

[9] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines". IEEE Transactions on Neural Networks, no. 13, pp. 415425, 2002.

[10] V. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.

[11] S. Knerr, L. Personnaz, and G. Dreyfus. "Single-layer learning revisited: A stepwise procedure for building and training a neural network". In Fogelman-Soulie and Herault, editors, Neurocomputing: Algorithms, Architectures and Applications, NATO ASI. Springer, 1990.

[12] Bao-Liang Lu; Kai-An Wang; Utiyama, M.; Isahara, H., "A part-versus-part method for massively parallel training of support vector machines". Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on , vol.1, no., pp.-740, 25-29 July 2004

[13] B.E. Boser, I.M. Guyon, V.A. Vapnik, "A training algorithm for optimal margin classifiers". In Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, 1992. ACM

[14] J. H. Friedman, "Another approach to polychotomous classification". Technical report, Stanford Department of Statistics, 1996. http://www-stat.stanford.edu/reports/friedman/poly.ps.Z.

[15] U. Kreel. "Pairwise classification and support vector machines". In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods: Support Vector Learning, pages 255268. MIT Press, Cambridge, MA, 1999.

[16] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. "Large margin dags for multiclass classification". In Proceedings of 12th NIPS conference, 2000.

[17] Ph. Refregier and F. Vallet. "Probabilistic approach for multiclass classification with neural networks". In Proceedings of International Conference on Artificial Networks, pages 10031007, 1991.

[18] D. Price, S. Knerr, L. Personnaz, and G. Dreyfus. "Pairwise neural network classifiers with probabilistic outputs". In G. Tesauro, D. Touretzky, and T. Leen, editors, Neural Information Processing Systems, volume 7, pages 11091116. The MIT Press, 1995.

[19] T. Hastie and R. Tibshirani. "Classification by pairwise coupling". The Annals of Statistics, 26(1):451471, 1998.

[20] E. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers", J. Machine Learning Res., vol. 1, pp. 113141, Dec. 2000.

[21] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition", in Proc. Seventh Eur. Symp. Artificial Neural Networks, Bruges, Belgium, 1999, pp. 219224.

[22] P. Clarkson and P.J. Moreno, "On the use of support vector machines for phonetic classification". ICASSP, 1999.

[23] T. F. Wu, C. J. Lin and R. C. Weng, "Probability Estimates for multiclass classification by pairwise coupling", Journal of Machine Learning Research, 5 pp. 975-1005, 2004.

[24] K. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? An empirical study". Proc. Multiple Classifier Systems (pp. 278–285), 2005.

[25] H. Bourlard, N. Morgan, 1994. "Connectionist Speech Recognition: A Hybrid Approach". Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[26] A. J. Robinson, M. M. Hochberg, and S. J. Renals, 1996. "The use of recurrent networks in continuous speech recognition". In C-H. Lee, F.K. Soong, and K.K. Paliwal, editors, Automatic Speech and Speaker Recognition, 233-258. Kluwer.

[27] G. D. Cook and A. J. Robinson, "The 1997 ABBOT system for the transcription of broadcast news", in Proc. DARPA BNTU Workshop, Lansdowne, VA, 1997.

[28] N. Strm, L. Hetherington, T. J. Hazen, E. Sandness, and J. Glass, "Acoustic modeling improvements in a segment-based speech recognizer", in Proc. IEEE ASR Workshop, Keystone, CO, Dec. 1999.

[29] C. Ma, M. Randolph and J. Drish, "A Support Vector Machines-based rejection technique for speech recognition". In Internat. Conf. Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, UT, USA, Vol. 1, pp. 381-384, 2001.

[30] Y. Benayed, D. Fohr, J.P. Haton, and G. Chollet, "Confidence measures for keyword spotting using support vector machines". In International Conference on Acoustics, Speech, and Signal Processing. Volume 1, Issue, 6-10 April 2003 Page(s): I-588 - I-591 vol.1.

[31] V Wan, S Renals, "Speaker verification using sequence discriminant support vector machines", Speech and Audio Processing, IEEE Transactions on, Vol. 13, No. 2. (2005), pp. 203-210.

[32] Applications of Support Vector Machines to Speech Recognition A. Ganapathiraju, J. E. Hamaker, J. Picone, IEEE Transactions on Signal Processing, vol. 52, no. 8, pp. 2348-2355, 2004.

[33] V. Venkataramani and W. Byrne, "Lattice segmentation and support vector machines for large vocabulary continuous speech recognition", Proc. ICASSP, pp.817820, 2005.

[34] J. Stadermann and G. Rigoll, "A hybrid SVM/HMM acoustic modeling approach to automatic speech recognition", in INTERSPEECH-2004 ICSLP, 2004, pp. 661 664.

[35] R. Solera-Urena, D. Martin-Iglesias, A. Gallardo-Antolin, C. Pelaez-Moreno, F. Diaz-de-Maria. "Robust ASR using Support Vector Machines". Speech Communication 49, 253-267, 2007.

[36] S.E. Kruger, M. Schaffoner, M. Katk, "Speech Recognition with Support Vector Machines in a Hybrid System", in Proceedings of Interspeech 2005.

[37] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers". In Advances in Neural Information Processing Systems 11. Cambridge, MA: MIT Press.

[38] P. J. Moreno, and R. Rifkin, "Using the fisher kernel method for web audio classification", ICASSP, 2000.

[39] N. Smith and M. Niranjan, "Data-dependent kernels in svm classification of speech patterns", in International Conference on Spoken Language Processing, 2000.

[40] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S.A. Solla. "Structural risk minimization for character recognition". Advances in Neural Information Processing Systems, 4:471479, 1992.

[41] B. Pellom, "Sonic: The University of Colorado Continuous Speech Recognizer". Technical Report TR-CSLR-2001-01, CSLR, University of Colorado, March 2001.

[42] R. Cole and B. Pellom. "University of Colorado read and summarized story corpus", Technical Report TR-CSLR-2006-03, University of Colorado, 2006.

[43] D. Bolanos, W. Ward, S. Van Vuuren, J. Garrido, Syllable Lattices as a Basis for a Childrens Speech Reading Tracker", in InterSpeech, Antwerp, Belgium 2007.

[44] F. Wessel, R. Schlter, K. Macherey, and H. Ney, "Confidence Measures for Large Vocabulary Continuous Speech Recognition", IEEE Transactions on Speech and Audio Processing, vol. 9, pp. 288298, March 2001.

[45] A. Ganapathiraju, J. E. Hamaker, J. Picone. "Applications of Support Vector Machines to Speech Recognition", IEEE Transactions on Signal Processing, vol. 52, no. 8, pp. 2348-2355, 2004.

[46] S. E. Krger, M. Schaffner, M. Katz, E. Andelic, A. Wendemuth. "Speech Recognition with Support Vector Machines in a Hybrid System". In Interspeech-2005 ICSLP, pp. 993996, 2005.

[47] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods", in Advances in Large Margin Classifiers. Cambridge, MA: MIT Press, 1999.

[48] J. Torgesen, R. Wagner, C. Rashotte. Test of Word Reading Efficiency: TOWRE. Austin, TX: PRO-ED, 1999.

[49] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.

[50] A. P. Dempster, M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm". Journal of the Royal Statistical Society, pp. 139, 1977.

[51] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm". SIAM Review 26, pp. 195202, 1984.

[52] S. E. Kruger, M. Schaffoner, M. Katk, E. Andelic, A. Wendemuth, "Mixture of Support Vector Machines for HMM based Speech Recognition", in proceedings of The 18th International Conference on Pattern Recognition (ICPR'06).

[53] M.D. Richard and R.P. Lippmann. "Neural network classifiers estimate Bayesian a posteriori probabilities", in Neural Computation no. 3, pp. 461-483,1991.

[54] H. Bourlard, H. Hermansky, and N. Morgan. (1996), "Towards increasing speech recognition error rates". Speech Communication 18(3), 205–231.

[55] L. Tth and A. Kocsor, "A segment-based interpretation of HMM/ANN hybrids", Computer Speech and Language archive Volume 21, Issue 3 (July 2007).

[56] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev and P. Woodland, *The HTK Book*, Cambridge University Engineering Department, 2001.

[57] R. Cole and B. Pellom. University of Colorado read and summarized story corpus. Technical Report TR-CSLR-2006-03, University of Colorado, 2006.

[58] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.

[59] Kai-Fu Lee, "Context-Dependent Hidden Markov Models for Speaker-Idependent Continuous Speech Recognition", IEEE Transactions on Acoustics Speech and Signal Processing, 38(4):599-609, April 1990.

[60] Z. Rivlin, A. Sankar and H. Bratt, "HMM State Clustering Across Allophone Class Boundaries", In Proc. Eurospeech '97.

[61] C. Dugast, P. Beyerlein and R. Haeb-Umbach, "Application of clustering techniques to mixture density modeling for continuous-speech recognition", In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on.

[62] S. J. Young, "The general use of tying in phoneme-based HMM speech recognizers", in Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing, 1992, pp. 569-572.

[63] K.-F. Lee and H.-W. Hon, (Nov. 1989), "Speaker-Independent Phone Recognition using Hidden Markov Models", IEEE Trans. Acoustics, Speech, and Signal Processing 37(11), 1641–1648.

[64] R. Schwartz et al., "Towards robust hidden Markov models", presented at the DARPA Workshop Speech Recognition, June 1988.

[65] S. J. Young and P. C. Woodland. "The use of state tying in continuous speech recognition". In Proc. EUROSPEECH, pages 2203-2206, Berlin, Germany, Sep. 1993. 15.

[66] X. Aubert, P. Beyerlein and M. Ullrich, "A Bottom-Up Approach For Handling Unseen Triphones In Large Vocabulary Continuous Speech Recognition", In Proceedings of ICSLP '96 (1996).

[67] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling", in Proc. ARPA Workshop on Human Language Technology, 1994, pp. 307312.

[68] I. Shafran and M. Ostendorf, "Acoustic model clustering based on syllable structure", Computer Speech and Language, vol. 17, no. 4, pp. 311328, 2003.

[69] C. Füegen and I. Rogina, "Integrating dynamic speech modalities into context decision trees", in Proc. ICASSP, Istanbul, 2000, vol. III, pp. 12771280.

[70] H. Bourlard, N. Morgan, C. Wooters, S. Renals, "CDNN: A Context Dependent Neural Network for Continuous Speech Recognition". In Proceedings of the International Conference on Acoustics, Speech and Signal Processing 1992, Vol. 2, pp. 349-352, San Francisco.

[71] H. Franco, M. Cohen, N. Morgan, D. Rumelhart, and V. Abrash, "Context-Dependent Connectionist Probability Estimation in a Hybrid Hidden Markov Model-Neural Net Speech Recognition System", Computer Speech and Language ,Vol 8. pp. 211-222, 1994.

[72] C. Williams and M. Seeger, , T. Leen, T. Dietterich, and V. Tresp, Eds., "Using the Nystrm method to speed up kernel machines", in Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2001, vol. 13, pp. 682688.

[73] A. Smola and B. Schlkopf, "Sparse greedy matrix approximation for machine learning", in Proc. 7th Int. Conf. Mach. Learn., Stanford, CA, Jun. 2000, pp. 911918.

[74] D. Achlioptas, F. McSherry, and B. Schlkopf, "Sampling techniques for kernel methods", in Advances in Neural Information Processing Systems, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, vol. 14, pp. 335342.

[75] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations", J. Mach. Learn. Res., vol. 2, pp. 243264, Dec. 2001.

[76] E. Osuna, R. Freund, and F. Girosi. "An improved training algorithm for support vector machines". In Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, pages 276285, Amelia Island, FL, USA, 1997a.

[77] J. C. Platt. "Fast training of support vector machines using sequential minimal optimization". In B. Scholkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods Support Vector Learning, pages 185208. MIT Press, Cambridge, MA, 1999.

[78] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.

[79] G. Schohn, D. Cohn: "Less is More: Active Learning with Support Vector Machines". Proceedings of ICML-00, 17th International Conference on Machine Learning (2000)

[80] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems". Neural Comput., vol. 14, no. 5, pp. 11051114, May 2002.

[81] S. Asharaf, M. Narasimha Murty: "Scalable non-linear Support Vector Machine using hierarchical clustering". ICPR (1) 2006: 908-911

[82] I. W. Tsang, J. T. Kwok, P. M. Cheung, "Core vector machines: Fast SVM training on very large datasets". Journal of Machine Learning Research, 6, 363-392.

[83] I. W. Tsang, A. Kocsor, J. T. Kwok, "Simpler core vector machines with enclosing balls". ICML 2007: 911-918.

[84] J.J. Godfrey, E.C. Holliman, and J. McDaniel. "SWITCHBOARD: Telephone speech corpus for research and development". In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, volume I, pages 517520, March 1992.

[85] I. W. Tsang, A. Kocsor, J. T. Kwok, LibCVM ToolKit, a library for http://www.cse.ust.hk/ ivor/cvm.html.

[86] S. Godbole, S. Sarawagi and S. Chackrabarti, "Scaling multi-class support vector machines using inter-class confusion". In Proceedings of ACM SIGKDD-2002.

[87] D. Bolanos and W. H. Ward, "Implicit State-Tying for Support Vector Machines Based Speech Recognition". *In Proceedings of Interspeech 2008*, September 22-26. Brisbane, Australia.

[88] D. Bolanos, W.H. Ward, B. Wise and S. Van Vuuren, "Pronunciation Error Detection Techniques for Childrens Speech". *In Proceedings of Interspeech 2008*, September 22-26. Brisbane, Australia.

[89] Sreenivasamurthy, N. and A. Ortega. "Towards Efficient and Scalable Speech Compression Schemes for Robust Speech Recognition Applications". in IEEE International Conference on Multimedia and Expo. 2000.

[90] Zheng, F. and J. Picone, "Robust Low Perplexity Voice Interfaces. 2001", Institute for Signal and Information Processing.

[91] S. Kapadia, V. Valtchev, and S. J. Young, "MMI training for continuous phoneme recognition on the TIMIT database", Proc. ICASSP 1993, vol. 2, pp. 491- 494.

[92] B.-H. Juang, W. Chou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition", IEEE Trans. Speech Audio Proc., Vol. 5, May 1997.

[93] P. C. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition", Computer Speech and Language, vol. 16, 2002. pp. 2547.

[94] D. Yu and Li Deng, "Large-Margin Discriminative Training of Hidden Markov Models for Speech Recognition", in International Conference on Semantic Computing, 2007. ICSC 2007.

[95] B. Wise and S. van Vuuren. "Choosing software gems to improve childrens reading", Perspectives, vol. 33, 3, pp. 34-38, 2008.

[96] B. Wise, T. Weston, L. Sessions, J. Tuantranonont, T. Tomczyk, L. Snyder, N. Sager, T. Struemph, G. Golding, N. Ngampatipatpong, S. van Vuuren. "Adaptation and Comprehension in ICARE", Poster presented at Annual Meeting of IES Investigators in reading, writing and learning and cognition, Institute of Education Sciences, Washington, DC, June 7, 2007. GOP (baseline)Cm Cf Psegments

[97] S.M. Witt, S.J. Young, "Phone-level Pronunciation Scoring and Assessment for Interactive Language Learning", Speech Communication, pp. 95-108, 2000.

[98] L. Neumeyer, H. Franco, V. Digalakis and M. Weintraub, "Automatic scoring of pronunciation quality", Speech Communication, pp. 83-93, 1999.

[99] J. Morris and E. Fosler-Lussier, "Conditional random fields for integrating local discriminative classifiers", IEEE Transactions on Audio, Speech and Language Processing, Vol. 16, No 3, March 2008.