SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION

By

Aravind Ganapathiraju

A Dissertation Proposal
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 1999

TABLE OF CONTENTS

LIST OF TABLES ii

LIST OF FIGURES iii

CHAPTER

I. INTRODUCTION 1

II. HIDDEN MARKOV MODELS - CURRENT TECHNOLOGY. 7

III. SUPPORT VECTOR MACHINES 17

IV. KERNEL-BASED DISCRIMINATION 32

V. PRELIMINARY EXPERIMENTS. 41

VI. PROPOSED WORK AND EXPERIMENTS. 52

VII. REFERENCES 57

LIST OF TABLES

TABLE Page

1. Performance of various SVM kernels on the Deterding vowel data. The results for the Gaussian note network were obtained from [60]. 41

2. Performance of SVMs on the SWB phone data.. 42

3. Best performance of SVMs in the frame classification experiment. Only the common confusable phone pairs were evaluated.. 45

4. Performance of the SVM/HMM hybrid system on the N-best rescoring task. The traditional word-internal HMM systems comes in at 49.8% WER. 49

LIST OF FIGURES

FIGURE Page

1. Schematic overview of a statistical speech recognition system 2
2. State-of-the-art performance on tasks of varying complexity starting with isolated words all the way up to unconstrained conversational speech. 5
3. A simple continuous density HMM structure which is used commonly in most speech recognition systems. 7
4. Definition of a linear hyperplane classifiers. SVMs are constructed by maximizing the margin. 18
5. Sample classification by the Royal Holloway SVM applet using a polynomial kernel. This data cannot be classified by a linear separating margin. This is interesting in the sense that SVMs handle multi-modal data effectively.. 20
6. Phone classification experiment organization. k-means clustering and normalization are nor necessary if the classifiers can be made to “behave” computationally. 43
7. Training data for the s and phones and the support vectors (using RBF kernels) are shown as data points with concentric diamonds and squares. 46
8. Test data for the s and f phones (note that the data is normalized) and was chosen to be easily separable. 47

FIGURE Page

9. Hybrid HMM/SVM system. SVMs are used exclusively to reorder N-best lists generated by HMMs. All alignments are done by the HMM system. 48

10. Histogram of SVM distances on the training set for phones “s” and “z”. Note the bimodal distribution corresponding to positive and negative examples. The class conditional distributions (each mode) is typically not Gaussian. 52

CHAPTER I

INTRODUCTION

One of the most complex and least understood properties of human perception is its ability to carve the world into various categories. The plethora of signals that pound our sensory system do not necessarily confuse the brain, rather are segmented into clean chunks of information related to various experiences we have had in the past [1]. How does our brain achieve this amazing performance? This is a cognitive scientists' perception of the question —"Is representation better than discrimination?". We face with this same situation in speech research. Before we start to delve into this issue, we need a brief overview of speech technology, which a few decades back many people probably had a glimpse of in "Star Trek".

Speech Recognition Background

The process of speech recognition in humans can be represented in a rather simplistic fashion as an interaction between distinct knowledge sources as shown in Figure 1. We say this is simplistic, because various experiments suggest that humans do not recognize speech this way. However this is a nice framework to cast the problem of speech recognition as one of statistical pattern recognition. The speech signal that the speaker articulates is sampled and is transformed to a feature domain which we think is more reasonable for recognition (instead of the sample data domain). The feature domain has traditionally been the frequency domain. However the most commonly used features

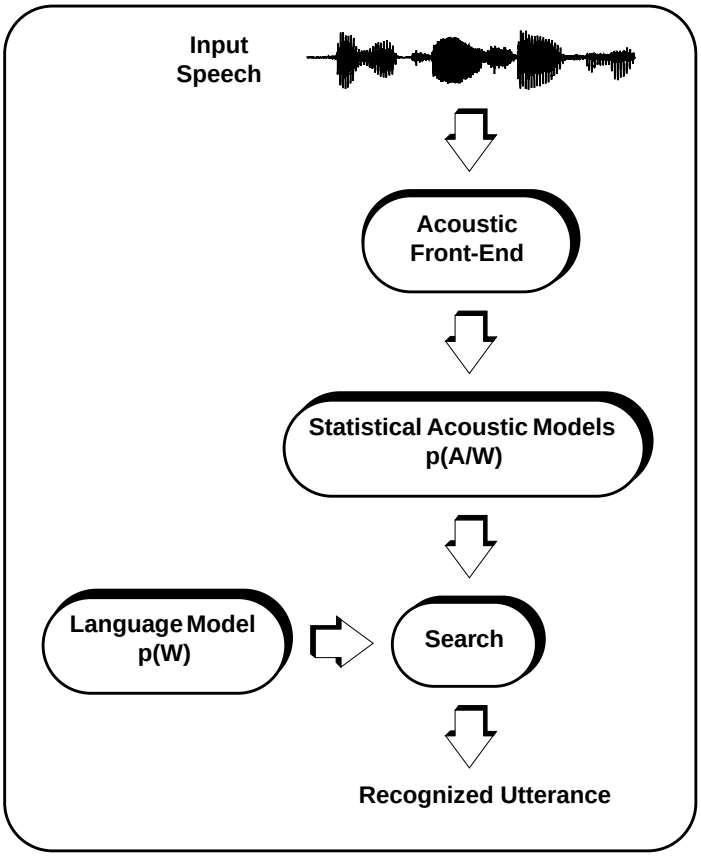

Figure 1. Schematic overview of a statistical speech recognition system

in current recognizers are in the cepstral domain [2,3,4,5,6]. This module of the speech recognizers typically called the Front-end. This is by the lowest level of knowledge that we incorporate into the system.

At a much higher level is the knowledge about the language that can be applied. The Language Model (LM), can take several forms. When the recognition tasks are small, the languages can be expressed as a set of re-write rules [7]. However when the tasks become more complex, one has to pose the LM in a more probabilistic framework. N-gram language models are one such commonly used methodology where the language is encoded as a probability distribution of a word given its predecessor words [6,8]. The

N-gram LM can take various forms depending on the representation of the history words — cache LMs, long-range LMs to name a few [6,8].

At the core of the recognizer is the Acoustic Model (AM). In the early days of speech research when the tasks being handled were relatively small vocabulary (a few hundred words), the AM typically represented the whole word and was used to model the various renditions of the word spoken by different people. As the recognition tasks became more complex, whole-word modeling became restrictive and systems transitioned to modeling sub-word units like phonemes and syllables [9,10,11,12,13,14].

Where does pattern recognition come into play? Once the input speech signal is converted into features, the system has to decode the underlying symbol sequence —the most likely symbol sequence to be precise. This mapping of features to the symbols is what makes a speech recognition and pattern recognition problem. The AM, to which the input data is compared to, must be robust enough to account for the natural variation in the articulatory process which varies from person to person. This variation is accounted for by estimating the parameters of the AM using information theoretic measures like Maximum Likelihood (ML) [16]. Some of the commonly used formulations for the pattern recognizers are Hidden Markov Models (HMM) [2,4,5,6] and Neural Networks [15,47,56,58].

Complexity of Recognition Tasks

As mentioned in the previous section, the tasks to which speech recognition is being applied to have changed significantly as the technology progressed. Recognition

tasks can be categorized based on several features.

- **Vocabulary Size**: This is by far the most important feature that differentiates the resource usage (memory and time) by a recognizer. On one end of spectrum is the two-word task of yes/no. On the other end is the task of automatically transcribing broadcast news where the vocabulary size could be as high as a few hundred thousand words.
- **Speaker Dependence**: One of the driving forces for current speech research is the possibility of giving speech interfaces to data retrieval tasks where users can get the data they need from a large database. By definition this warrants the speech interface to be speaker-independent. On the other hand speech dictation systems on our personal computers are designed to perform well as they learn the speaking characteristics of the user. The jump in modeling complexity when we go from speaker-dependent systems to speaker-independent systems is very significant.
- **Recording Conditions**: The effectiveness of the features extracted by the front-end is dependent to a large extent on the inherent noise in the raw data. This noise could have been added to the speech signal as part of the recording system or the ambience. For example features extracted from speech recorded over the telephone have to deal with the degradation imposed by the telephone channel bandwidth and echo. Speech recognition of data recorded via the cellular network has to cope with severe ambient noise. On the other extreme, early speech research was based on speech recorded in a sound room with

controlled ambience.

- **Style of Speech**: Another important issue while comparing various recognition tasks (and part of the problem this thesis is trying to address) is the style of speech. Early speech research started with recognizing isolated words where there is a clear pause between utterances. With continuous speech like news broadcasts, the systems need to deal with artifacts like coarticulation and lack of clear segmentation between words. However this can still be considered "read speech". With conversational speech systems need to start dealing with non-speech sounds, interjections, restarts and a slew of other disfluencies [18,19].

Figure 2 shows the performance degradation on typical tasks that cover the gamut of present speech recognition systems. Though there has been a tremendous increase in

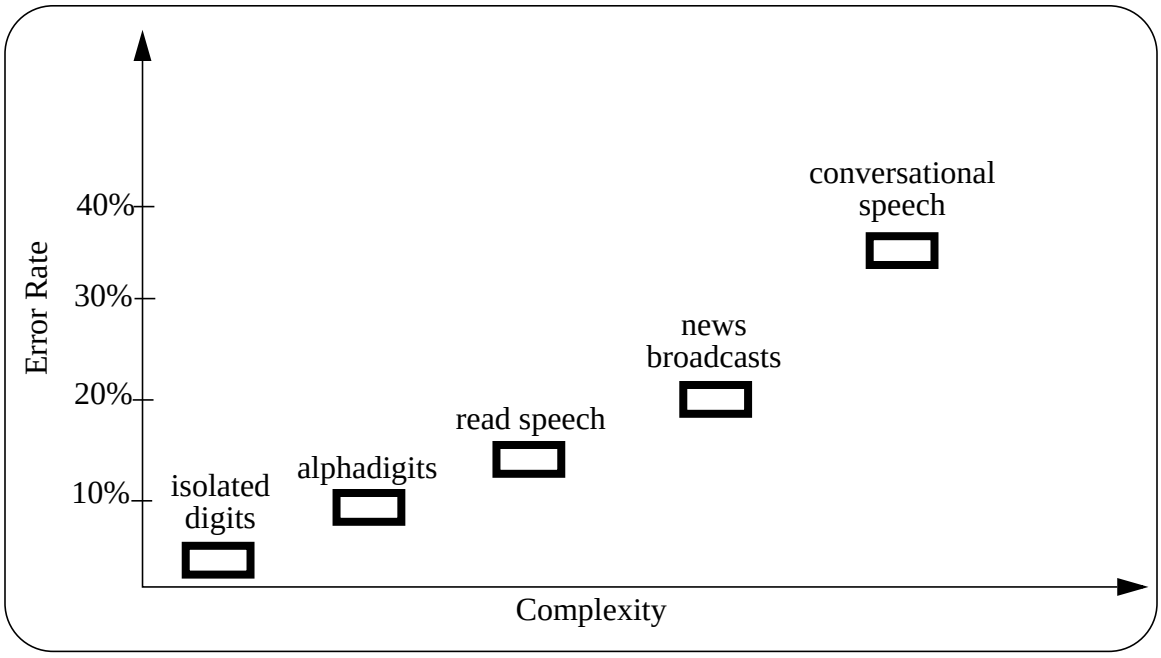

Figure 2. State-of-the-art performance on tasks of varying complexity starting with isolated words all the way up to unconstrained conversational speech.

compute power over the years, the trend in the plot clearly indicates a need to solve several problems in core technology before the systems become more prevalent.

CHAPTER II

HIDDEN MARKOV MODELS - CURRENT TECHNOLOGY

Hidden Markov models (HMM) provide an elegant mathematical framework to represent the time sequential nature of speech as well the variability in the speech sounds. HMMs are by definition represent a doubly stochastic process where one stochastic process represents the temporal behavior (via state sequences) and the other process represents the variability in speech [2,4,5,6]. At the core of the HMM is a Bayes classifier where classification is done using a simple likelihood ratio. Other classifiers like Multi-Layered Perceptrons (MLP) and Support Vector Machines (SVM) differ from HMMs in their inability to model temporal evolution of speech [21,44]. HMMs can use discrete or continuous output probability distributions depending on the input symbol set. In this thesis only continuous density HMMs will be described.

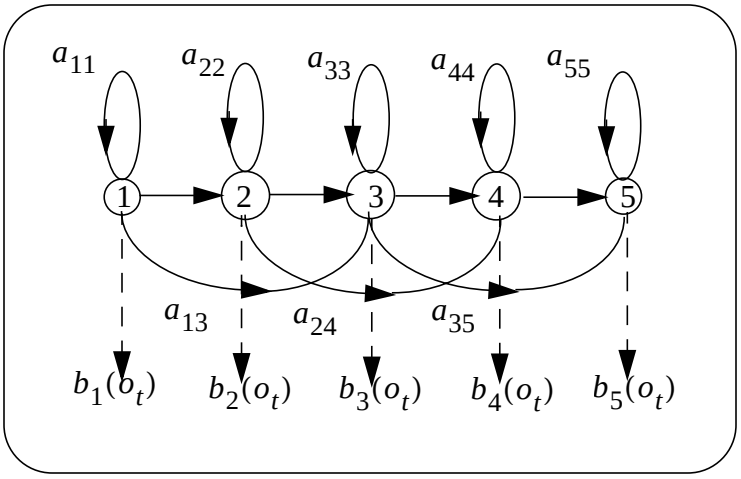

Figure 3. A simple continuous density HMM structure which is used commonly in most speech recognition systems.

Basic Definition

HMMs are finite state machines in their basic form. They differ from regular finite state machines in that each state also has a probability of emitting a symbol. Apart from this there is a probability distribution representing the probability of a transition from one state to another [4]. The complete description of the model can be provided using the following quantities:

- N — the number of states
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(\boldsymbol{o})\}$, where $\boldsymbol{o}$ is the input observation vector

The output probability distribution gives the probability of observing a vector in the given state. The most commonly used form of the output distribution is a multivariate Gaussian. Other distributions like Laplacians have been used in some systems [20]. The multivariate Gaussian can be written as:

$$b_j(\boldsymbol{o}_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} \exp\left(-\frac{1}{2}(\boldsymbol{o}_t - \mu_j)'\Sigma_j^{-1}(\boldsymbol{o}_t - \mu_j)\right), \quad (1)$$

where $\boldsymbol{o}_t$ is the observation vector at time t and the subscript j indicates that the Gaussian under consideration belongs to the jth state. Figure 3 shows an example of a five state HMM with skip transitions. A few terms need to be defined in order to better understand the use of HMM in recognizing speech. The following formulation assumes

that the input consists of T observation vectors.

Forward Probability

The forward probability gives us the probability of generating the observations from time 1 to t and the model ending in state j at time t.

$$\alpha_j(t) = Pr(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t, x(t) = j) \tag{2}$$

The above computation can be efficiently done using the following recursive formulation.

$$\alpha_j(1) = a_{ij} b_j(\boldsymbol{o}_1) \tag{3}$$

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\boldsymbol{o}_t), \text{ for } 1 < t \leq T \text{ and,} \tag{4}$$

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T) a_{iN} \tag{5}$$

Backward Probability

The backward probability is the probability of generating the observations from time $t+1$ to T if the model was in state j at time t.

$$\beta_j(t) = Pr(\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T \,|\, x(t) = j) \tag{6}$$

Similar to the forward probability computation, a recursive formulation exists for the backward probability computation.

$$\beta_i(t) = a_{iN} \tag{7}$$

$$\beta_i(t) = \sum_{i=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1), \text{ for } 1 \le t < T \text{ and,} \tag{8}$$

$$\beta_1(1) = \sum_{i=2}^{N-1} a_{1j} b_j(\boldsymbol{o}_1) \beta_j(1) \tag{9}$$

Utterance Likelihood

The total utterance probability, P, given the model M, can be written in terms of α and β as:

$$P(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_j(t) \beta_j(t), \tag{10}$$

where $\boldsymbol{O}$ is the sequence of observation vectors.

Assumptions and Limitations in HMMs

Though the HMM paradigm is very attractive for speech recognition, several assumptions need to be made about the structure of the process (speech in this case) that they are modeling [29]. Some of these assumptions blatantly violate our knowledge of the speech production process.

Stationarity

For the speech production process to be modeled by a state machine we need to

assume that during the period the model is in a particular state the complete process can be represented by one feature vector. This implies that we assume stationarity for the duration of a frame of speech data. Typically we use a frame analysis rate of 20-100 frames/second.

Independence of Observations

In the HMM, the likelihood of generating an observation is dependent only upon the state and is independent of all other observations. This is not true of speech tends to change slowly compared to the frame rate which makes observations dependent on one another. The very fact that the human speech production system is a complex dynamical system incapable of sudden transitions, makes this assumption significant. To allow the correlation between data from adjacent frames, most speech recognition systems augment the input feature vectors with their derivatives.

State Transition Probability

From the definition of the transition probabilities between states in an HMM, we note that the probability of staying in a state decreases exponentially with time. This is not something we see in real data where the state distributions take the form of a Gamma distribution. The work around this problem is to explicitly model state duration probabilities along with the observation probabilities [22]. This has however not proven to be effective.

Parameter Estimation

The goal of the HMM parameter estimation process is to maximize the likelihood of the data given the model, traditionally known as Maximum Likelihood (ML) estimation [6,16,17]. In effect ML tries to maximize the a posteriori probability of the training data given the model. Note that this implies that other models are not part of this optimization process. One of the most compelling reasons for the success of ML and HMMs has been the existence of iterative methods to estimate the parameters while guaranteeing convergence. Expectation-Maximization (EM) is one algorithm that is used extensively to perform ML estimation.

EM Theorem and Maximum Likelihood

If,

$$\sum_t P_{\theta'}(t|y)\log P_{\theta}(t|y) > \sum_t P_{\theta'}(t|y)\log P_{\theta'}(t|y) \tag{11}$$

then,

$$P_{\theta}(y) > P_{\theta'}(y) \tag{12}$$

The gist of the above theorem is that, if we start with a model θ' and find a model θ such that equation 11 is satisfied, then the observed data y is more probable under the model θ than under θ' [6]. This is a very powerful theorem in that it guarantees convergence at least to a local minimum. In the above formulation t is the intermediate random variable that depends on the model parameter settings. For example, t could be

the state sequence in an HMM which is not something we observed. The terms on the LHS and RHS of equation 11 can be represented as the auxiliary functions $Q(\theta', \theta)$ and $Q(\theta', \theta')$. Since we are maximizing the auxiliary function in the EM framework, the parameter update equations can be obtained by differentiating $Q(\theta', \theta)$ with respect to each of the parameters and setting the derivative to zero. When t, is chosen as the state sequence, the EM formulation is called the Baum-Welch algorithm [23]. A detailed explanation of the update equations for each of the HMM parameters can be found in [24].

The ML approach works well if the assumption that the form of the parametric probability model that computes $P(O|M)$ is the same as the true underlying distribution. This is a very restrictive assumption is many cases and assumes the availability of a large amount of training data to estimate the parameters of a complex process like speech.

Another drawback of the ML approach is that the model parameters are estimated based on the data belonging to that model only. It is independent of all the other models being estimated. This is however not the best way to improve recognition performance. Some form of discrimination needs to be added to the estimation process to improve the performance. MLPs and SVMs estimate parameters discriminatively. However they are not tractable to model temporal variation in their basic form. The next section describes a couple of commonly used discriminative training techniques, at the core of which is still an HMM parameter optimization problem.

Discriminative Estimation Techniques

The primary difference between HMM parameter estimation via ML and other discriminative techniques is that the objective criterion in the latter includes the probability of the data given that the wrong model was used [24,25,26]. This provides the optimization process with the negative examples of the data being generated by the model. The motivation for discriminative techniques could be either based on some information theoretic concept or directly trying to minimize the ultimate goal of speech recognition — reduced classification error [28].

Maximum Mutual Information

The mutual information, I, between variables X and Y is defined as the average amount of uncertainty about the knowledge of X given knowledge of Y [17]. Mathematically this can be defined as:

$$I(X;Y) = H(X) - H(X/Y) \tag{13}$$

The conditional entropy of X given Y is given by

$$\sum_{x,\,y} P(x,\,y)\log P(x/y) = -E[\log P(x/y)] \tag{14}$$

Having defined mutual information, we now pose the speech recognition problem in the same framework. Let W, O denote the random variables corresponding to the words and observation vectors. The uncertainty in the word sequence given the acoustic observations is the conditional entropy of W given O,

$$H(W/O) = H(W) - I(W;O). \quad (15)$$

Note that we do not know $P(w, o)$ in general and need to estimate a parametric fit. The conditional entropy of the words given the acoustic observations can be shown to confirm to the following inequality:

$$H_{\lambda}(W/O) \geq H(W/O), \quad (16)$$

where λ denotes a particular parametric fit to the actual distribution [17,24,26]. The equality holds only if $P_{\lambda}(w/o) = P(w/o)$. Thus by minimizing 15, we can get an optimal estimate of the conditional distribution. This minimization can also be done as a maximization of the mutual information and hence the name maximum mutual information (MMI) to this optimization technique.

The objective function L_{MMI} for the MMI estimation of the parameters is nothing but the mutual information of the words given the acoustic observations under the parametric distribution λ.

$$L_{MMI}(\lambda) = I_{\lambda}(X;Y) = H_{\lambda}(W) - E[\log P_{\lambda}(w/o)] \quad (17)$$

Replacing the expectations by the sample averages and assuming the training data to comprise of R utterances,

$$L_{MMI}(\lambda) = -\frac{1}{R}\sum_{r=1}^{R}\log P_{\lambda}(w_r) + \frac{1}{R}\sum_{r=1}^{R}\log\frac{P_{\lambda}(o_r|M_r)P_{\lambda}(w_r)}{P_{\lambda}(o_r)} \quad . (18)$$
$$= \frac{1}{R}\sum_{r=1}^{R}\{\log P_{\lambda}(o_r|M_r) - \log P_{\lambda}(o_r)\}$$

In the above equation w_r is the word sequence in the r^{th} utterance with a corresponding composite model M_r. o_r are the set of observation vectors corresponding to the r^{th} utterance. The first term in the above equation is equivalent to the ML optimization criterion and the second term is what makes this a discriminative framework.

CHAPTER III

SUPPORT VECTOR MACHINES

Empirical risk minimization is one of the most commonly used optimization procedures in machine learning. In this regime, the goal is to arrive at a parameter setting that gives the smallest value for,

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}\left|y_i - f(x_i, \alpha)\right| \tag{19}$$

where α is the set of adjustable parameters. Note that this risk computation is done over the training examples only. Neural network training (back-propagation in particular) is a direct consequence a similar optimization process. There are no probability computations involved in the above definition of risk. Another form of risk commonly used is,

$$R(\alpha) = \int\frac{1}{2}|y - f(x, \alpha)|\,dP(x, y) \tag{20}$$

This is the estimated risk. Vapnik proved that bounds exist for this expected risk such that,

$$R(\alpha) \leq R_{emp}(\alpha) + f(h), \tag{21}$$

where h is called the Vapnik Chervonenkis (VC) dimension [21,32,37]. In general we cannot compute the left hand side of 21. However, if we know h, we can compute the right hand side. Thus when we are given several learning machines with a given empirical

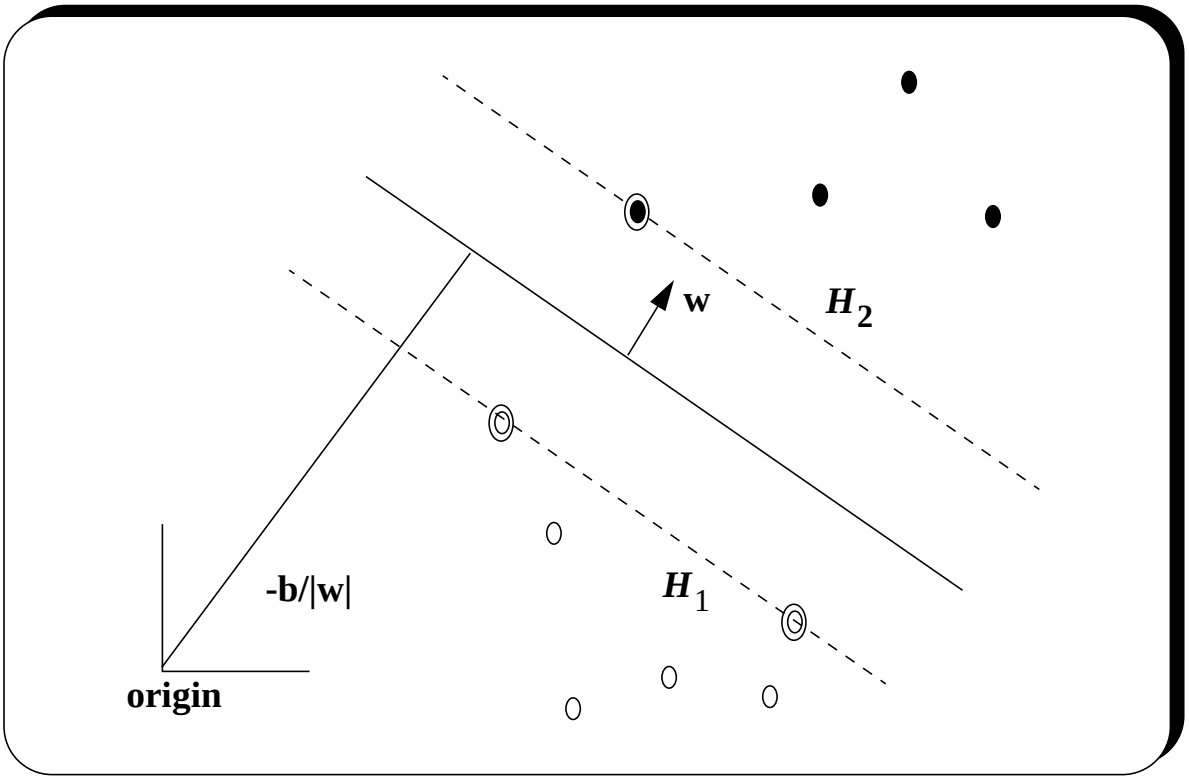

Figure 4. Definition of a linear hyperplane classifiers. SVMs are constructed by maximizing the margin.

risk, we can choose the machine that minimizes the right hand side. This principled approach is called Structural Risk Minimization [21]. Support Vector Machine (SVM) theory is built on top of this founding principle.

Linear Hyperplane Classifiers

The power of SVMs lies in transforming data to a high dimensional space and constructing a linear binary classifier in this high dimensional space. Construction of a hyperplane in a feature space requires transformation of the n-dimensional input vector $\boldsymbol{x}$ into an $\boldsymbol{N}$-dimensional feature vector, i.e.

$$\Phi:(\Re^n \rightarrow \Re^N) \tag{22}$$

A N-dimensional linear separator $\boldsymbol{w}$ and a bias b are then constructed for the set of transformed vectors. Classification of an unknown vector $\boldsymbol{x}$ is done by first transforming the vector to the feature space and then computing

$$\operatorname{sgn}(\boldsymbol{w} \cdot \boldsymbol{x} + b). \tag{23}$$

The above formulation is based on the fact that among all hyperplanes separating the data, there exists a unique one that maximizes the margin of separation between the classes. Figure 4, shows a typical 2-class classification example where the examples are perfectly separable using a linear decision region. H_1 and H_2 define two hyperplanes the distance between which needs to be maximized. Let $\boldsymbol{w}$ be the normal to the decision region. Let the N training examples be represented as the tuples $\{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$ where $-1 \leq y_i \leq 1$. The points that lie on the hyperplane, that we assume, to separate the data satisfy,

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \tag{24}$$

where b is the distance of the hyperplane from the origin. Let the "margin" of the SVM be defined as the distance between closest positive and negative example from the hyperplane. The SVM looks for the separating hyperplane which gives the maximum margin. Once the hyperplane is obtained, all the training examples satisfy the following inequalities.

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 \qquad \text{for } y_i = +1 \tag{25}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 \qquad \text{for } y_i = -1 \cdot \tag{26}$$

The above equations can be compactly represented as a single inequality,

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0 \qquad \forall i \tag{27}$$

Looking at the above equations with respect to Figure 4, we see that all points satisfying 25 lie on H_1 and the points satisfying 26, lie on H_2. The distance between H_1 and H_2 is $2/\|\boldsymbol{w}\|$. Note that for a completely separable data set, no points fall between H_1 and H_2. Thus for maximizing the margin, we need to minimize $\|\boldsymbol{w}\|^2$. The reason we choose the square of the norm is that there are elegant techniques to optimize convex functions with constraints. The training points for which the equality in 27 holds are called Support Vectors. In Figure 4, they are indicated by concentric circles. So, now we face with a minimization problem given a set of inequality constraints. The theory of

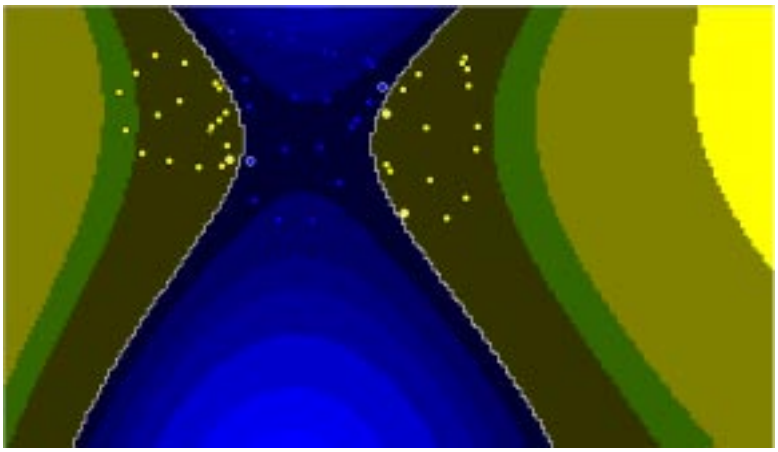

Figure 5. Sample classification by the Royal Holloway SVM applet using a polynomial kernel. This data cannot be classified by a linear separating margin. This is interesting in the sense that SVMs handle multi-modal data effectively.

Lagrange multipliers has been in existence for such problems for several decades [34].

The functional for the above optimization problem, called the Lagrangian, can be written as,

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{N} \alpha_i \tag{28}$$

This is clearly a convex quadratic programming problem since the objective function itself is convex. The above is called the primal formulation. Since we are minimizing L_P, its gradient with respect to $\boldsymbol{w}$ and b should be equal to zero. This gives the conditions,

$$\boldsymbol{w} = \sum_i \alpha_i y_i x_i, \text{ and} \tag{29}$$

$$\sum_i \alpha_i y_i = 0. \tag{30}$$

Substituting the above equations into 28, we get the dual formulation,

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{31}$$

SVM learning can thus be treated as the problem of maximizing L_D with respect to α_i subject to their positivity and the constraints 30.

In constrained optimization problems with inequality constraints Karush-Kuhn-Tucker (KKT) are necessary and sufficient conditions for a solution to

exist [34].

Karush-Kuhn-Tucker Theorem [31,34]

Let $f, \boldsymbol{h}, \boldsymbol{g}$ be three functions. Let x^* be a regular point and a local minimizer for the problem of minimizing f subject to $\boldsymbol{h}(x) = 0$, $\boldsymbol{g}(x) \leq 0$. Then there exist $\lambda^* : \in \Re^m$ and $\mu^* : \in \Re^p$ such that:

$$\mu^* \geq 0 \tag{32}$$

$$Df(x^*) + \lambda^{*T} D\boldsymbol{h}(\boldsymbol{x^*}) + \mu^{*T} D\boldsymbol{g}(\boldsymbol{x^*}) = 0^T, \text{ and} \tag{33}$$

$$\mu^{*T} \boldsymbol{g}(\boldsymbol{x^*}) = 0. \tag{34}$$

Using the third of the KKT conditions, we get

$$\alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1) = 0 \tag{35}$$

Equations 29 and 35 are the solutions for $\boldsymbol{w}$ and b respectively. Thus far we have seen the case where the training data is completely separable using a linear margin. However, we know very well that this is not the case with real-world data. Most of the classification problems involve non-separable data. Given such a training set, we still need to get the best classifier possible. The optimal-margin classifier can be extended to this non-separable case by using a set of slack variables. In this situation, the inequality constrains become,

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 - \xi_i \qquad \text{for } y_i = +1, \tag{36}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 + \xi_i \qquad \text{for } y_i = -1, \text{ and} \tag{37}$$

$$\xi_i \geq 0 \qquad \forall i. \tag{38}$$

A close look the above inequalities shows that for an error to occur, the corresponding ξ_i needs to be greater than 1. This implies that the upper bound on the number of errors on the training data is $\sum_i \xi_i$. The optimization process in the new data setting needs to minimize this quantity also. The new term that is added to the objective function is $C\left(\sum_i \xi_i\right)^2$, where C is used to control the penalty for a training error. The solution of the optimization now includes the constraint,

$$0 \leq \alpha_i \leq C \tag{39}$$

The higher this value the harder the optimization process will try to minimize training errors. However this could mean increased time for convergence and in some cases a larger support vector set.

Non-linear Hyperplane Classifiers

Now that we have seen the problem of estimating linear classifiers for cases where data is both separable or non-separable. This however does not help us solve many real-world situations where the data warrants the need for non-linear decision surfaces.

This section deals with extending the SVM paradigm to handle such cases.

In all the formulations of the optimization in the previous sections notice that the only place the data points occur is in the dot product. Suppose the data points are transformed to a higher dimension using

$$\Phi : \Re^d \rightarrow \Re^D, \tag{40}$$

where D is the dimensionality of the new feature space. In this new space we can still construct optimum margin classifiers with the only difference being that the simple dot product ill now have to be replaced by $\Phi(x_{i'}) \cdot \Phi(x_j)$. If we can define a "kernel" function K that could compute this dot product for us using x_i and x_j as inputs, then we circumvent the need to know the form of Φ explicitly. In this new formulation, out decision function will take the form,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i \Phi(x_{i'}) \cdot \Phi(\boldsymbol{x}) + b. \tag{41}$$

Of course there are several issues one has to take into account before we can guarantee the existence of the kernel K. Mercer's theorem can be used to ascertain that the pair $\{H, \Phi\}$ exists for a given kernel where H defines the new feature space [37].

Mercer's Theorem

There exists a mapping Φ and an expansion

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \Phi_i(\boldsymbol{x}) \cdot \Phi_i(\boldsymbol{y}), \tag{42}$$

if and only if, for any $g(x)$ such that

$$\int g(\boldsymbol{x})^2 dx \text{ is finite,} \tag{43}$$

then

$$\int K(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{x}) g(\boldsymbol{y}) dx dy > 0 \tag{44}$$

So, the Mercer's conditions tells us if a kernel is a dot product in some space. It does not however directly tell us how we define Φ or even the feature space H [21,32,36]. Some of the commonly used kernel functions are:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d \text{ — polynomial with degree } d \tag{45}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = Sigmoid[s(\boldsymbol{x} \cdot \boldsymbol{y}) + c] \text{ — sigmoid} \tag{46}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\{-\Upsilon|\boldsymbol{x} - \boldsymbol{y}|^2\} \text{ — radial basis function} \tag{47}$$

SVM Training Process

Though a solution for the quadratic optimization is guaranteed, the number of computations required can get very high depending on the separability of the data and the number of training data points. Several heuristics need to be considered to make SVM training possible is a reasonable amount of time. This section will discuss some of the important modifications to the optimization problem formulation that make handling tasks

with thousands of training points and support vectors possible.

Optimizing the functional in 31 can be very expensive if the data set is large (over a few thousand training samples) and may not fit the physical memory even on most modern day computers. Choosing the right optimizing regime becomes that much more important in this case. Several approaches have been developed, based on some nice properties of the functional and the constraints, to make this optimization possible.

Chunking

This method is based on the idea of dividing the optimization problem into sub-problems whose solution can be found efficiently. This method divides the training data into chunks and optimizes the functional for each chunk. How does this guarantee overall optimization? Osuna proves that the chunking algorithm does in fact give the same solution as a global optimization process but takes much less operating memory and time [38,39].

The functional to optimize, 31 and the constraints on the Lagrange multipliers can be written in a vector form as:

$$\boldsymbol{W}(\Lambda) = -\Lambda \cdot \mathbf{1} + \frac{1}{2}\Lambda \cdot D\Lambda, \text{ subject to,} \tag{48}$$

$$\Lambda^{T} \cdot \boldsymbol{y} = \mathbf{0} \tag{49}$$

$$\Lambda - \boldsymbol{C}\mathbf{1} \leq \mathbf{0} \text{ and} \tag{50}$$

$$-\Lambda \leq \mathbf{0}. \tag{51}$$

In the above defined form of the functional,

$$D_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{52}$$

The above problem needs to satisfy the KKT conditions to guarantee optimality. For constraints 49, 50 and 51, let the KKT multipliers be μ, Υ and Π. The KKT conditions for this problem, based on 32, 33 and 34, are

$$\begin{aligned} \nabla W(\Lambda) + \Upsilon - \Pi + \mu \boldsymbol{y} &= \mathbf{0} \\ \Upsilon^T (\Lambda - C\mathbf{1}) &= 0 \\ \Pi^T \Lambda &= 0 \\ \Upsilon &\geq \mathbf{0} \\ \Pi &\geq \mathbf{0} \end{aligned} \tag{53}$$

Now we can simplify the above conditions to a simple form based on the range into which a Lagrange multiplier λ_i falls.

1. $0 < \lambda_i < C$: Since the multiplier is non zero, π_i and ν_i should be zero to satisfy the KKT conditions. This results in the following equation.

   $$(D\Lambda)_i - 1 + \mu y_i = 0 \tag{54}$$

2. $\lambda_i = C$: This value for the multiplier implies that π_i is zero, leading to $(D\Lambda)_i - 1 + \nu_i + \mu y_i = 0$. If we define $g(\boldsymbol{x}_i)$ as the estimated label value given by

   $$g(\boldsymbol{x}_i) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \tag{55}$$

then, $y_i g(\boldsymbol{x}_i) = 1 - \nu_i$. Since ν_i is positive,

$$y_i g(\boldsymbol{x}_i) \leq 1 \tag{56}$$

The support vectors that have their multipliers at C are called Bounded Support Vectors (BSV) and are important in learning the inherent overlap in the data.

3. $\lambda_i = 0$: This condition implies that ν_i is zero and hence, $(D\Lambda)_i - 1 + \pi_i + \mu y_i = 0$ Following a procedure similar to the previous case it can be shown that for the multiplier in this range,

$$y_i g(\boldsymbol{x}_i) \geq 1 \tag{57}$$

The conditions 56 and 57 are essential in deciding whether a multiplier is violating the KKT conditions which in turn will be used to choose the best set to optimize so that the overall optimization is fast. The Chunking algorithm can be specified in three simple steps [38]. Suppose we define the working set as B and the non-working set (whose multipliers do not change while solving the sub-problems) as N.

1. Choose $|B|$ training points from the data set at random.
2. Solve the optimization problem defined by the set B.
3. For some $j \in N$, such that:
   - $\lambda_j = 0$ and $y_j g(\boldsymbol{x}_j) < 1$
   - $\lambda_j = C$ and $y_j g(\boldsymbol{x}_j) > 1$

- $0 < \lambda_j < C$ and $y_j g(\boldsymbol{x}_j) \neq 1$

replace and λ_i, $i \in B$, with λ_j and the solve the new sub-problem given by:

Minimize $W(\Lambda_B) = -\Lambda_B^T \mathbf{1} + \frac{1}{2}\Lambda_B^T \cdot D_{BB}\Lambda_B^T + \Lambda_B^T \boldsymbol{q}_{BN}$, subject to, (58)

$$\begin{aligned} (\Lambda_B^T \cdot \boldsymbol{y}_N + \Lambda_N^T \cdot \boldsymbol{y}_N = 0) \\ \Lambda_B - C\mathbf{1} \leq 0 \\ -\Lambda_B \leq 0 \end{aligned} \tag{59}$$

where,

$$(\boldsymbol{q}_{BN})_j = y_j \sum_{i=1}^{N} \lambda_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{60}$$

The above algorithm is guaranteed to strictly improve the objective function based on the observations made by Osuna [38]. The convex quadratic form of the objective function also guarantees that the algorithm will reach the global optimum solution within a finite number of iterations. The challenge now is find the best working set at each iteration and to device heuristics to avoid redundant computations. The work by Joachims addresses this issue of the SVM training algorithm [42].

Some Observations

A fallout of having BSVs as a solution to a sub-problem is that they can be given low priority for being considered for choosing the next chunk of data. In fact in some

cases where we know that the training data is mislabelled, we may decide to throw away the BSVs from the optimization process altogether. The very fact that they are at the upper bound implies that they are maximally violating the KKT conditions. If they are not handled carefully, they may end up becoming a support vector in the solution for every sub-problem.

If, SVM optimization is a solved problem, why are they not more commonly used? The answer to this question is the — “It depends on the data”. The most important aspects that users need to consider before using SVMs for an application are:

- Nature of the data: SVMs by definition are static classifiers. In order to handle data that evolves with time and has features varying with time, we need to look at hybrid methods
- Size of the data set: The larger the data set, the more the number of computations involved in the optimization process. Though new algorithms like Chunking have made SVMs capable of handling large data sets, this was one of the main bottlenecks in the past.
- Separability of the data: If the training data is separable, SVMs will learn the decision region quickly and efficiently (in terms of the number of SVs that are needed to represent the decision region). With separable data, training could be a potential problem and most of the training vectors may end up becoming SVs. This makes SVM based classification very inefficient. We do have some control over the allowed errors in the training data with the use of the parameter C.

- Number of classes involved: Since SVMs are inherently two-class classifiers, they would require N classifiers in order to handle an N-class problem. The resource usage in this case is much higher compared to other techniques like, decision trees [43], HMMs or neural networks.

CHAPTER IV

KERNEL-BASED DISCRIMINATION

The last two chapters saw us develop the theory behind HMMs and SVMs. We have also looked at some of the pros and cons involved in both the machine learning techniques. This chapter probes into one possible way to tie HMMs and SVMs in order to get better performance. This method is motivated by the fact the HMMs do a good job modeling the characteristics of a given class and SVMs learn the difference between this class and any other class.

Likelihood Ratio Tests

In HMM based continuous speech recognition, the output distributions in each state of the HMM could be either discrete or continuous. In the case of continuous distributions, Gaussians are typically used to model the probabilities. The most common computation in this setup involves the probability of the input sequence given the model. When we need to determine which model was more likely to have generated the input feature sequence, we do a likelihood ratio test, typically converting the probabilities to the log. domain for computational ease [15,44].

Let $O = \{o_1, o_2, \ldots, o_2\}$, be the observation sequence (cepstral coefficients for example). The aim is to find if model M_1 or M_2 is more likely to have produced the observation sequence. In the probability space, this can pose as a likelihood ration test as:

$$L = \log\frac{P(O/M_1)P(M_1)}{P(O/M_2)P(M_2)} = \log\frac{P(O/M_1)}{P(O/M_2)} + \log\frac{P(M_1)}{P(M_2)} \quad (61)$$

The first term in the above equation is the classic definition of the maximum likelihood criterion which is the premise on which HMMs are built. The second term is the a priori information and is ignored in most cases assuming a uniform prior. Thus the decision process simply involves looking at the sign of L. When L is positive, we choose M_1 as the model that generated O and vice-versa.

The parameters of the HMM are however estimated to maximize $P(O/M_1)$ and $P(O/M_2)$ individually using only the positive examples of data generated by M_1 and M_2 respectively. In reality however it would be better if we could estimate the probabilities by forcing the model to be more likely for positive examples and less likely for negative examples. In chapter 2, we did see one such framework while discussing MMIE estimation of HMM parameters.

Equation 61 can be rewritten as,

$$L = \log\frac{P(M_1/O)P(O)}{P(M_2/O)P(O)} = \log\frac{P(M_1/O)}{P(M_2/O)}, \quad (62)$$

where $P(O)$ is the probability of the input observations. $P(M_1/O)$ and $P(M_2/O)$ are the posterior probabilities of O belonging to the models M_1 and M_2 respectively. The goal of the parameter estimation should be to optimize these posteriors

directly. ANNs do this inherently in their estimation process though the parameter estimation is actually done via the least-squares method [15]. With SVMs, this needs to done explicitly by converting SVM distances to probabilities [46,59].

Kernels and Sufficient Statistics

From chapter 3, we saw that kernel methods provide a measure that can be used to compare any two input sequences unlike the likelihood in a traditional HMM that only gives a measure of closeness of the sequence to the model itself. In fact it may happen in many cases that two different HMMs may give the same likelihood to two completely different input sequences. This can happen because each of the HMMs was estimated in isolation of the other. This section will try to develop the theory to build an SVM on top of the underlying HMMs. This mechanism will hopefully allow us to use the representative power of the HMMs (as well as their ability to handle temporal variations) in conjunction with the discriminative power of SVMs.

In order to describe the training data, all algorithms based on some form of stochastic optimization will need the gradient of an objective function with respect to the parameters of the HMM. It is the solution to these gradients that gives us the optimal estimated value. The vector of sufficient statistics describes the process of generating the underlying state sequence from the HMM. For example, the quantities in this vector would include the posterior probabilities of taking a particular state transition or emanating a particular observation vector for a state. In traditional HMM parameter estimation, the log likelihood of the data given the model is the objective function. Thus any observation

sequence can be converted to a fixed length feature vector comprising of the sufficient statistics. Instead of dealing with the sufficient statistics directly, we can work with the gradients described above. These scores are also called the Fisher Scores [44]. Fisher scores are part of the commonly used definition for the Fisher Information Matrix [45]. The Fisher matrix is often viewed as the information contained in the data set about each parameter.

HMMs parameter estimation is based on the maximization of $P(O/M)$ where M is the parameter set that defines the model and needs to be optimized. The Fisher information matrix I, for this case is defined as,

$$I = E_O\left\{U_O U_O^T\right\}, \text{ where} \tag{63}$$

$$U_O = \nabla_M \log P(O/M) \tag{64}$$

and the expectation is over $P(O/M)$. U_O is called the Fisher score. We can define a distance metric for two observation sequences O_i and O_j mapped to this model M as,

$$K(O_i, O_j) = U_{O_i} I^{-1} U_{O_j} \tag{65}$$

in terms of the Fisher scores. The right-hand side of the above equation is positive definite and hence can be kernel as per Mercer's conditions [21]. Note how the above definition of a kernel ties in our notion of HMM parameter estimation with kernels that

forma an integral part of SVM theory. The Fisher kernel defined above can be used as any other kernel defined in the previous chapter to determine a separating hyperplane in the Fisher score space or a higher dimensional space. This method will perform at least as well as the underlying HMM model [44]. We now have a simple procedure where the advantages of the HMM model are available to the discriminative classifier, SVM, via the Fisher kernel. With logistic regression models, I^{-1} can be viewed as the covariance matrix of a Gaussian prior and is often discarded from 65. We do not however know if that is true in the case of traditional HMMs. This aspect will be explored further as part of the proposed research.

Computation of Fisher Scores

Having seen the definition of Fisher kernels we now move on to defining the Fisher scores in terms of the HMM parameters. We will see how these scores can be easily obtained with minor modifications to the Baum-Welch computations [24]. For simplicity we will derive the required quantities assuming single mixture Gaussian models with diagonal covariances.

Let us first define some of the terms we will use in defining estimates of all the constituents of the Fisher score vectors. Let $\boldsymbol{O}$ be the observation sequence and let M be the models under consideration with a parameter set λ. The likelihood of the observation sequence given the model is defined as,

$$L_{\lambda}(O/M) = \log P_{\lambda}(O/M) \tag{66}$$

such that,

$$\frac{\partial L_\lambda}{\partial \lambda} = \frac{1}{P_\lambda(\boldsymbol{O}|M)} \frac{\partial}{\partial \lambda} P_\lambda(O/M) \tag{67}$$

The posterior probability can be written in terms of the α and β as,

$$P_\lambda(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_j(t)\beta_j(t).$$

In order to introduce the other parameters in the model into the above equation, we can write it as,

$$P_\lambda(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} b_j(\boldsymbol{o}_t)\beta_j(t) \tag{68}$$

Transition Probability

The transition probabilities need to be handled carefully in order to guarantee that the transitions out of any state sum to unity. For this reason, a regularization function is used to redefine the transitions as,

$$a_{ij} = \frac{f_a(h_{ij})}{\sum_k f_a(h_{ik})} \text{ and } f_a(x) = e^x \tag{69}$$

which is also known as softmax is some literature [15]. Then,

$$\frac{\partial a_{ij}}{\partial h_{ik}} = a_{ij}(\delta_{kj} - a_{ik}), \tag{70}$$

where δ is the kroneker delta.

Since P depends on a_{ij}'s, applying the chain rule gives,

$$\frac{\partial}{\partial h_{ik}} P_\lambda(O/M) = \sum_j \frac{\partial}{\partial a_{ij}} P_\lambda(O/M) \frac{\partial a_{ij}}{\partial h_{ik}}. \tag{71}$$

Differentiating 68 with respect to a_{ij} and using it along with 70 in 71 yields,

$$\frac{\partial}{\partial h_{ik}} P_\lambda(O/M) = \sum_t \sum_j \alpha_i(t-1) a_{ij}(\delta_{kj} - a_{ik}) b_j(\boldsymbol{o}_t) \beta_j(t) \tag{72}$$

The above equations used in conjunction with 67 provide means to get the components in the Fisher score vector corresponding to the transition probabilities.

Mean

In order to get the contribution of the means towards the Fisher scores, we need to start with the definition of a Gaussian as in 1. Differentiating this equation with respect to the d^{th} component of the mean of distribution corresponding to the j^{th} state, we get,

$$\frac{\partial}{\partial \mu_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t) \left\{ \frac{o_{td} - \mu_{jd}}{\sigma^2_{jd}} \right\} \tag{73}$$

From 68, we know that,

$$\frac{\partial}{\partial b_j(\boldsymbol{o}_t)} P_\lambda(O/M) = \sum_{t=1}^{T} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} \beta_j(t) \tag{74}$$

Using the chain rule for partial derivatives, we get,

$$\frac{\partial}{\partial \mu_{jd}} P_\lambda(O/M) = \sum_{t=1}^{T} C(t, j) b_j(\boldsymbol{o}_t) \left\{ \frac{o_{td} - \mu_{jd}}{\sigma^2_{jd}} \right\}, \text{ where,} \tag{75}$$

$$C(t, j) = \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} \beta_j(t) \tag{76}$$

Using 75 and 76 in 67 gives the contribution of the mean vectors toward the Fisher scores.

Variance

As mentioned earlier, for simplicity we assume diagonal covariances in these derivations. In the case of diagonal covariances, we need to constrain the values such that all of them are positive. In order to convert the constrained set to an unconstrained set (as we did with the transitions), we use the following regularization.

$$\sigma_{jd}^2 = f(z_{jd}) \text{ and } f(x) = e^x \tag{77}$$

We can start first by looking at the gradient of the output distribution with respect to the variance.

$$\frac{\partial}{\partial \sigma^2_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t)\frac{1}{2}\left\{\frac{(o_{td}-\mu_{jd})^2}{(\sigma^2_{jd})^2} - \frac{1}{\sigma^2_{jd}}\right\} \tag{78}$$

Using 77, we can convert 78 in terms of the regularization variable z as,

$$\frac{\partial}{\partial z_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t)\frac{1}{2}\left\{\frac{(o_{td}-\mu_{jd})^2}{\sigma^2_{jd}} - 1\right\} \tag{79}$$

Using 74 and the chain rule for partial derivatives, we get,

$$\frac{\partial}{\partial z_{jd}} P_\lambda(O/M) = \sum_{t=1}^{T} C(t, j) b_j(\boldsymbol{o}_t)\left\{\frac{(o_{td}-\mu_{jd})^2}{(\sigma^2_{jd})^2} - 1\right\} \tag{80}$$

Using 80 in 67, we get the contribution of the variances towards the Fisher score.

The above quantities are also used in other discriminative techniques like MMIE training and MCE estimation [24,25,26,28]. These quantities can be easily extended to include multiple instances of the model and multiple mixture Gaussians.

CHAPTER V

PRELIMINARY EXPERIMENTS

Since SVMs have proven to be effective on classical pattern recognition problems [], a logical first step in this work was apply this technique to classification of phonetic segments in speech at various levels. The main aim of the set of experiments described in this chapter is to ascertain that SVMs can in fact provide good discrimination of phone segments. This would then be followed by integrating this paradigm into a hybrid SVM/HMM system, similar to hybrid ANN/HMM systems that have had significant success in the past [15,48]. All the SVM experiments described in this proposal were done using the publicly available SVM toolkit, SVM*Light* [42]. The system was modified at places to accommodate the requirements of the experiments described here.

Some of the key features in SVMLight include:

- fast optimization algorithm based on Chunking described in the previous chapter
- caching of kernel evaluations for speed

order/gamma/ hidden-units	RBF	Polynomial	Gaussian Node Network
2/0.025/22	32	**42**	46
3/0.05/88	**31**	44	47
4/0.1/528	32	45	**45**

Table 1. Performance of various SVM kernels on the Deterding vowel data. The results for the Gaussian note network were obtained from [60].

- handles many thousands of support vectors
- handles several ten-thousands of training examples
- supports standard kernel functions and lets you define your own

All experiments involving ANNs were performed using another publicly available toolkit, NICO [49]. This toolkit has been used successfully for simple classification experiments as well as more complex applications involving hybrid HMM/ANN technology. This toolkit supports recurrent connections as well as time-delay windows which are essential for speech recognition. The topology of the networks can be user defined via a simple user interface.

All HMM based experiments were done using the ISIP ASR system [55].

Vowel Classification

In our first pilot experiment, we applied SVMs to a publicly available vowel classification task [47]. In this evaluation, the speech data was collected at a 10 kHz sampling rate and low pass filtered at 4.7 kHz. The signal was then transformed to 10 log-area parameters, giving a 10 dimensional input space. A window duration of

Order/Gamma	RBF	Polynomial
2/0.1	68	68
3/1	64	66
5/10	**51**	68

Table 2. Performance of SVMs on the SWB phone data.

50 msec. was used for generating the features. The training set consisted of 528 frames from eight speakers and the test set consisted of 462 frames from the remaining seven speakers. Table 1 shows the performance of two type of kernels on the test data. Performance using both the kernels is better than most nonlinear classification schemes [46]. The best performance reported on this data set is, however, 29% error using a speaker adaptation scheme called Separable Mixture Models [50]. Neural network classifiers (Gaussian Node Network) produce a misclassification rate of 45% [47].

Phone Classification

In our next experiment, 16 phones were extracted from selected utterances in the Switchboard Corpus. The phones were chosen to represent vowels, the fricatives ‘s’ and

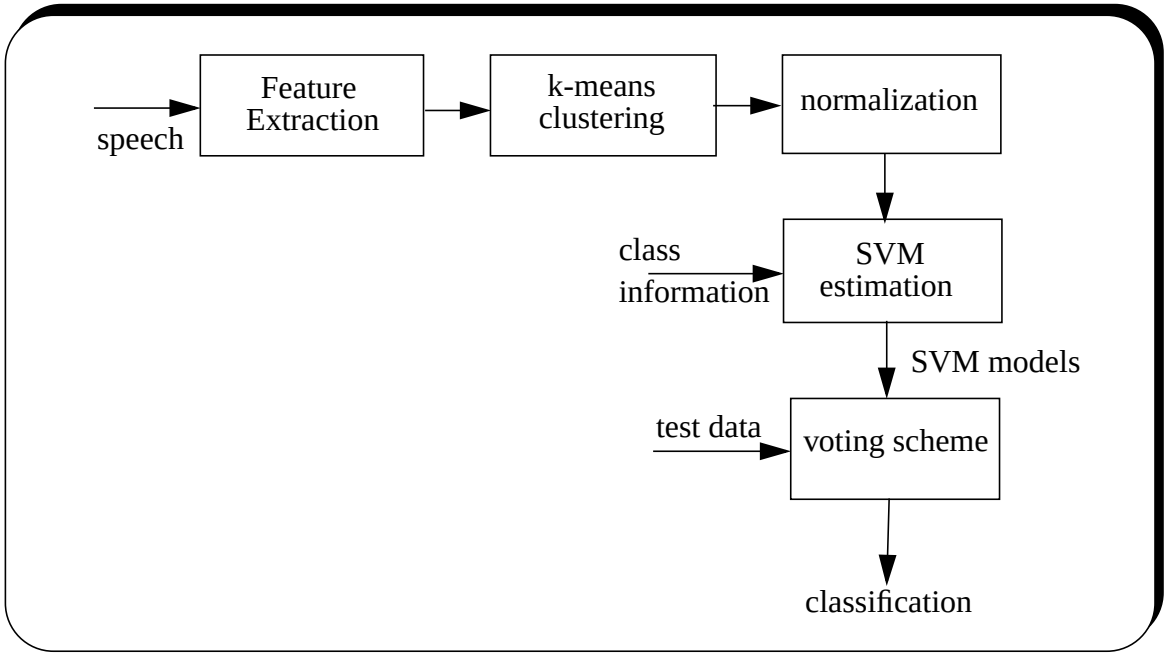

Figure 6. Phone classification experiment organization. k-means clustering and normalization are nor necessary if the classifiers can be made to “behave” computationally.

‘f’ and the liquids ‘l’ and ‘r’ [51]. The segmentation was based on a 44 phone context-independent HMM system where each phone was modeled as a 3-state HMM. As exemplars of each phone, we chose only the data that was mapped to the central state of the corresponding HMM. Feature vectors were generated by computing 12 mel-scaled cepstra along with energy. A frame duration of 10 msec. and a window duration of 25 msec. was used for data generation. Though in most speech recognition systems we also use the derivatives and the acceleration counterparts of the base features, we decided to keep the data simple for these preliminary experiments.

To avoid dealing with problems associated with the optimization process involved in training the database, we clustered the data for each phone into 200 clusters using 5000 exemplars. A simple k-means algorithm with a mean-squared error distance measure was used for the clustering process. However, to avoid clusters representing features with large average values, we normalized the features to a [-1, +1] range before the clusters were generated. The test set was chosen from the normalized data to represent a speaker independent portion. It consisted of 100 exemplars per phone to a total of 1600 test vectors.

Table 2 shows the classification results obtained by using RBF and polynomial kernels. Though these misclassification rates are on the higher side, it is worth noting that the phone error rate on this data is about ~68% and one would expect a higher frame mis-classification rate. Using a higher dimensional feature vector, which either includes the derivatives and acceleration coefficients or uses concatenated frames could improve the performance significantly. Experiments along these lines have been using in ANN

systems with proven success [56,58]. Since dealing with high dimensional data is not a concern with SVMs, the above technique will be studied as part of the proposed work.

Frame Classification

The previous set of experiments were performed on very highly confusable data. In order for us to better analyze the performance of SVMs, we performed experiments on a smaller task, Alphadigits. The OGI Alphadigit corpus [52] is a telephone database collected using a T1 interface with over 3000 subjects reading a list of either 19 or 29 alphanumeric strings (e.g., “8 h a 8 b h”). All experiments were performed on a training set of 51545 utterances from 2237 speakers and evaluated on 3329 utterances from 739 speakers [54].

Its acoustic properties are similar to SWB corpus described below. The 1102 unique strings comprising the prompted utterances were each six words long, and each list was designed to balance the phonetic context of all word pairs.

phone pair	**SVM misclassification rate**	**HMM misclassification rate**
f <=> sil	14.6	13.1
r <=> l	11.9	17.8
s <=> sil	37.5	42.4
s <=> z	9.7	17.8
t <=> p	8.7	18.1
t <=> d	9.6	22.2

Table 3. Best performance of SVMs in the frame classification experiment. Only the common confusable phone pairs were evaluated.

The first experiment involved phone classification using single mixture HMM models. The primary motivation for doing this was to get a confusion matrix for phones so that the highly confusable pairs would be analyzed using SVMs. The most confusable phone pairs included, “s -> z”, “s -> f”, “t -> p” and “t -> d”. Notice these are the classic minimal pairs encountered in most telephone speech data.

As part of the analysis we looked at the HMMs for the phones “s” and “f”. We visually inspected the Gaussians in the trained models to find the dimensions with most discrimination and confusion. Dimensions 1 and 2 (cepstral coefficients 1 and 2) were the most confusable. Figure 7 shows the incredible amount of confusion in these dimensions.

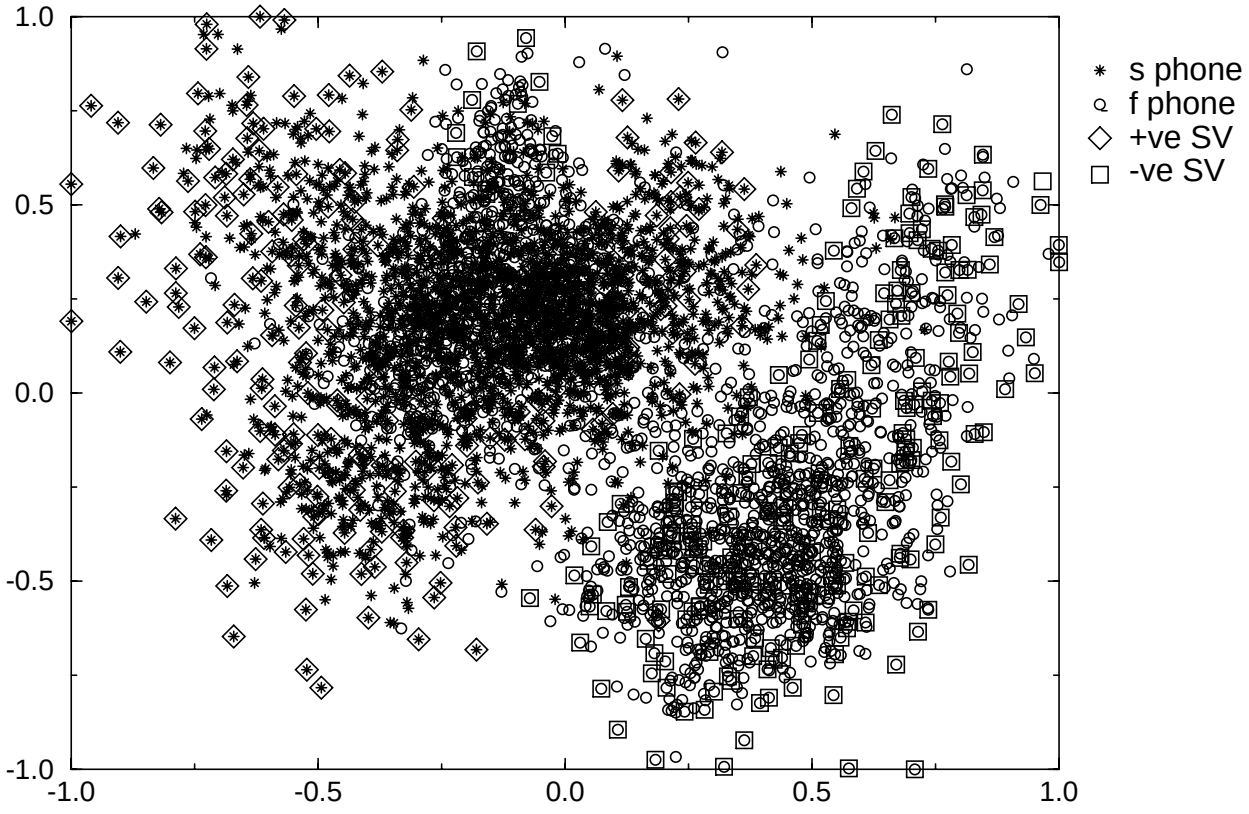

Figure 7. Training data for the s and phones and the support vectors (using RBF kernels) are shown as data points with concentric diamonds and squares.

Notice the number of overlapped f samples in the s region. We chose a test set that was relatively clean (Figure 8). This was done in order to quantify the effect of the overlapped data. The first experiment done using SVMs failed in that the frame misclassification rate was close to 40%. The training data clearly indicates why this would be the case with SVMs. The SVMs were trying to add all the overlapped data into the support vector set.

In chapter 3 we defined a set of support vectors that were at the bounds called BSVs. All the support vectors that were trying to model the overlapped data fell into this class. During optimization of the SVMs, we can throw away all the BSVs as cases of inconsistent examples. This is as amazing feature of the SVMs — automatic data

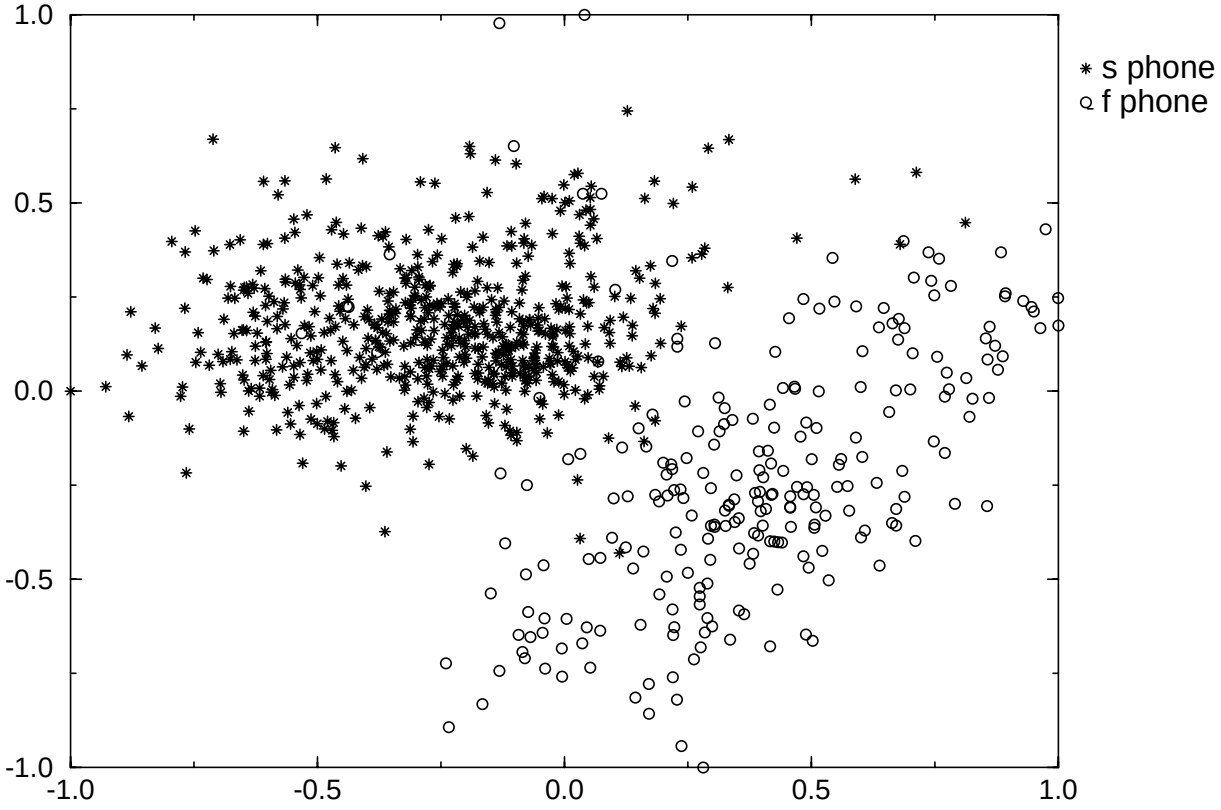

Figure 8. Test data for the s and f phones (note that the data is normalized) and was chosen to be easily separable.

cleaning. We then trained the SVMs using the above property. In Figure 7, notice that there are no SVMs trying to model the overlapped data for the negative, i.e. f phone, examples. This system gave an frame mis-classification rate of 3%. In order to compare with other classification schemes, namely HMMs and neural networks, we constructed the same and tested on the same test set. The neural network system, a simple MLP with one hidden layer with 32 hidden units, gave a 9% mis-classification rate. The HMM system built using the above training data came in at 14% mis-classification rate. Having learnt the effect of BSVs on training, we trained the SVMs forcing BSVs to be removed from the optimization process. With this setup, SVMs outperformed HMMs on almost all confusion pairs (except for f<=>sil) as shown in table 3. The difference is performance is very significant and encouraging.

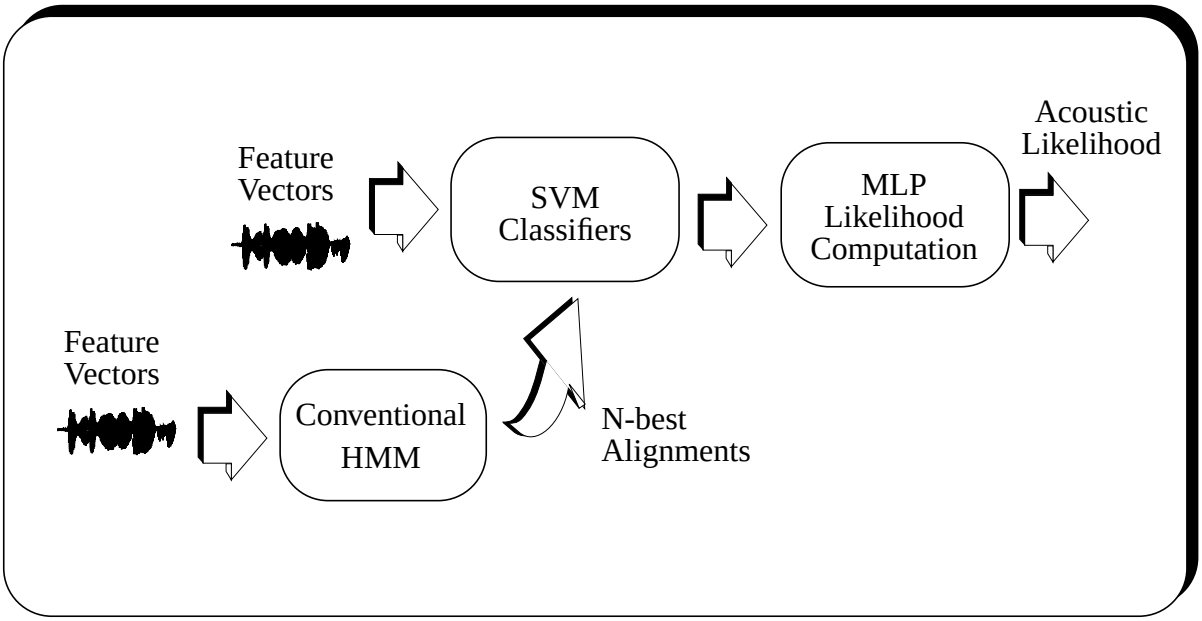

Figure 9. Hybrid HMM/SVM system. SVMs are used exclusively to reorder N-best lists generated by HMMs. All alignments are done by the HMM system.

Rescoring Experiments

In the last few sections we have seen clear evidence of the classification power of SVMs even while dealing with highly confusable data. Of course most speech recognizers are not simply frame level classifiers. They use a hierarchical approach to recognition starting at the frame level and moving all the way up to the sentence level [55]. Any new machine learning technique needs to be integrated into this framework in order to be able to do a fair comparison with HMMs.

As a first step towards this integration process, we designed a simple setup where we use information from conventional HMM systems. We then add information provided by the SVMs to get to final solution. This setup is shown in Figure 9. Assuming that we have already trained the SVM classifiers, we start with N-best lists generated by a conventional HMM system (a word-internal triphone system in this case).

The hybrid system we built was a context-independent system with 42 phone

N-Best	SVM Rescore
1	—
5	48.5%
10	49.8%
15	52.4%
20	55.8%
25	55.4%
30	54.5%

Table 4. Performance of the SVM/HMM hybrid system on the N-best rescoring task. The traditional word-internal HMM systems comes in at 49.8% WER.

classes (one classifier for each). The model level alignment for each of the hypothesis in the N-best list was generated using a 32 mixture context-independent HMM system. Based on these alignments the frames are classified using the 42 SVMs. However, SVMs output distances and not posterior probabilities. In order to convert the distances to posterior probabilities, a simple MLP with a softmax layer was used. The output of the MLP is a measure of the probability though only an approximation. These probabilities are used to compute the utterance likelihood of each hypothesis in the N-best list. The N-best list is reordered based on the likelihood and the top hypothesis is used to compute the word error rate (WER).

Table 4 shows the performance of this HMM/SVM hybrid system on N-best lists for a standard test set of the Switchboard corpus [53]. This test set was developed at the 1997 Johns Hopkins Workshop [10]. These results are very interesting, especially the performance on the 5-best lists. The word-internal context dependent system used to generate the N-best lists gave a WER of 49.8% on the test set. The 1.3% reduction in WER using the hybrid system is significant and encouraging.

There are however several issues that need to be addressed as part of the proposed work. One of the most compelling questions that arise here is — Is the confusable phone set the same for HMMs and SVMs?. If the answer is yes, then the experiment done here is valid. However, if the answer is no, then the N-best lists do not mean much for the SVM system. The hybrid system could give more gains if the confusion pairs in the N-best lists matched those for SVMs. Another aspect that has not been addressed here is the time alignments. In real hybrid systems the time alignments should be obtained as part of the

decoding process.

CHAPTER VI

PROPOSED WORK AND EXPERIMENTS

The previous chapters described the theory behind HMMs and SVMs and have provided motivation to use SVMs in conjunction with HMMs. Results on preliminary experiments described in the previous chapter are very encouraging. However, they only addressed the pattern classification component of the research. We need to perform a slew of experiments to ascertain that hybrid HMM/SVM systems do indeed give significant

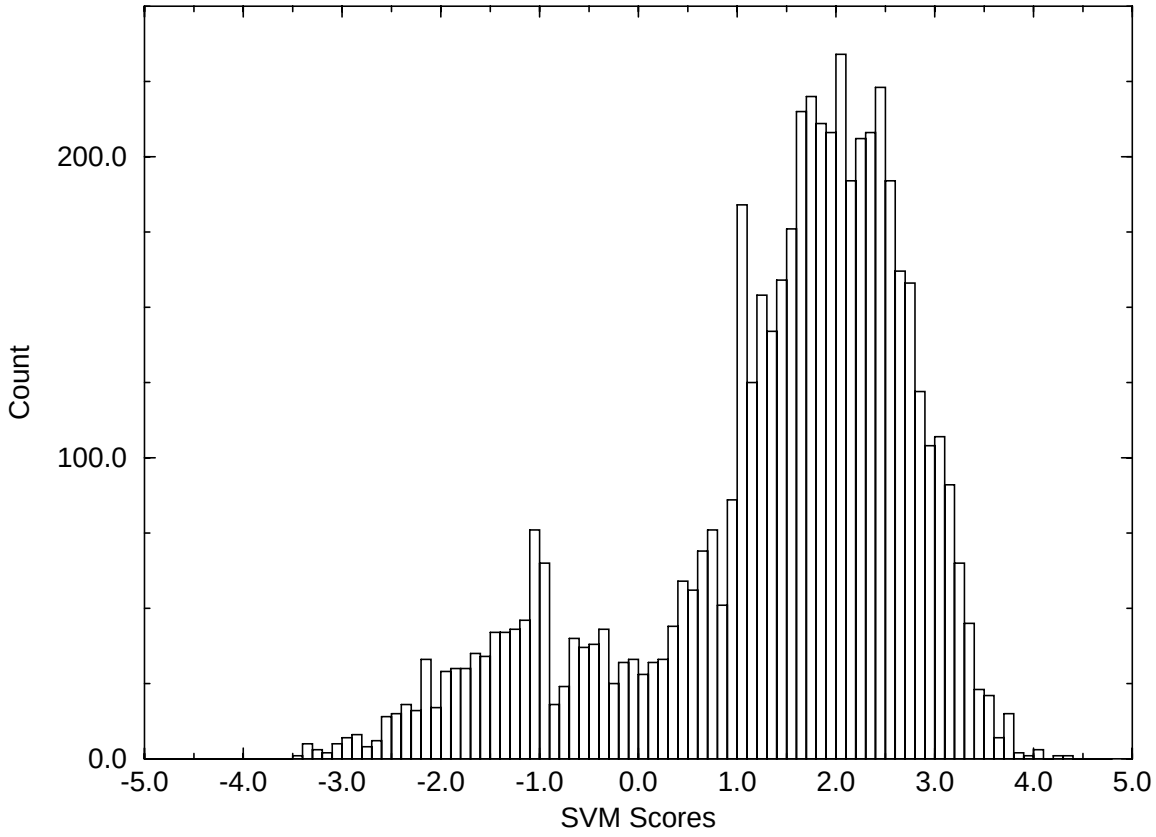

Figure 10. Histogram of SVM distances on the training set for phones “s” and “z”. Note the bimodal distribution corresponding to positive and negative examples. The class conditional distributions (each mode) is typically not Gaussian.

improvement over traditional HMM based systems. This chapter describes experiments towards this end.

Training and Evaluation Data

The primary datasets that will be used for evaluating the work proposed here will be the OGI Alphadigit Corpus and the Switchboard (SWB) corpus [52,53]. For the alphadigit data all experiments will be performed on a training set of 51545 utterances from 2237 speakers and evaluated on 3329 utterances from 739 speakers [54]. For the SWB based experiments we will use the training and test sets created during the 1997 LVCSR Summer workshop at Johns Hopkins University [10]. The training set consists of 60 hours of speech or 114000 utterances. The test set consists of 2427 utterances and is an open-loop set, i.e. no speakers from the training set appear in the test set.

Frame Classification

In the last chapter we described preliminary frame classification experiments performed on alphadigit data. These experiments clearly indicate the power of SVMs in learning decision regions on highly confusable data. This will be extended to a more challenging task, SWB. Using a conventional HMM system a phone confusion matrix will be constructed and the most confusable pairs of phones will be evaluated using SVMs. The performance of the system will be compared with the simple Bayes classifier and MLPs.

Another aspect of the machine learning techniques being compared here is their behavior towards input of dimension other than 13 cepstral coefficients augmented by

their derivatives and acceleration coefficients. In experiments not reported here, we found that the performance of HMMs using only the base features (no gradients) is significantly worse than while using the gradients. In order to build simple systems, this aspect of machine learning techniques will be studied in this work. An interesting observation from the experiment mentioned above is that HMMs using acceleration coefficients along with the base features does better than the system using base and delta features. This is an indication of the need for a wider context in the acoustic space. Similar to experiments done with MLPs, we will investigate the effect of using feature vectors comprising of multiple frames of data on the classification rate with SVMs.

N-best Rescoring Experiments

We have previously described our first attempts at integrating SVMs into a real speech recognition system via an N-best rescoring paradigm. However, these experiments did not explore all the possible avenues. Notably absent was a clean method to convert SVM distances to likelihoods. Significant work on this topic has been done in the past couple of years. ANNs have a clear advantage over SVMs in this respect because of the ease with which the outputs from the ANNs can be converted to posterior probabilities [56,57,58]. One simple way of handling this situation is to fit a Gaussian for the class conditional probabilities $p(f/y = 1)$ and $p(f/y = -1)$. The posterior probability $p(y = 1/f)$ will then be a sigmoid [59]. Unfortunately the Gaussian assumption for the class-conditional probabilities is not typically true as shown in Figure 10. In closer look at the figure tells us that any symmetric distribution is not

appropriate for this case since the larger the distance from the margin the better it is. In terms of a distribution this would however mean that the farthest examples are the least probable for that class.

Platt suggests that instead of fitting the class-conditionals first, it is better to parametrically fit the posteriors directly [21]. The posterior takes the form,

$$p(y = 1 / f) = \frac{1}{1 + \exp(Af + B)} \tag{81}$$

The free parameters A and B are estimated using the training data and simple prior estimates. In this work I plan on using the above method and incorporating it into the rescoring framework. It has also been seen that using a cross-validation scheme is better than estimates both the posteriors and the SVM parameters using the same set (especially for non-linear kernels [59]).

Kernel Discrimination Experiments

As described in chapter 4, the key component of kernel discrimination are the Fisher scores. These scores can be obtained while training the HMMs using the forward backward algorithm. The recipe to use this system is as follows:

Training

- Perform a Viterbi alignment on the training utterances in order to identify all phone instances and their time information
- compute the fisher scores using a pass of forward-backward algorithm
- train the Fisher SVMs to discriminate between the positive and negative

examples

Testing

- use N-best lists to start with
- For each utterance in the N-best list do a pass of forward-backward algorithm in order to compute the Fisher scores for the models
- Compute the posterior probabilities of the phones based on the SVM classifications and generate one probability for the complete utterance.
- Reorder the N-best lists using the probabilities generated in the previous step.

The above experiments will be first performed on alphadigit data followed by SWB. Note also that all experiments above will be done using context-independent HMM models. With context-dependent models we need to be worried about the number of classes that need to be classified (typically there are about 8000 models in a SWB system). A possible way to avoid this explosion is to follow a strategy followed in hybrid HMM/ANN systems where classifiers are built to first predict the context. These probabilities can then be used in conjunction with classifiers for context-independent models to simulate a context-dependent framework [15]. This methodology will be explored as part of the proposed work.

The baseline systems that will be used for comparing the systems described above will be context-dependent triphone systems for both alphadigits and SWB. These systems perform at 9.7% and 45.6% WER respectively on the test sets described previously.

CHAPTER VII

REFERENCES

[1] S. Harnad, *Categorical Perception: The Groundwork of Cognition*, Cambridge University Press, New York, 1997.

[2] J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing, New York, USA, 1993.

[3] S. Furui, "Cepstral Analysis Technique for Automatic Speaker Verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 29, No. 2, pp. 254-272, April 1981.

[4] J. Picone, "Continuous Speech Recognition Using Hidden Markov Models," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 7, no. 3, pp. 26-41, July 1990.

[5] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[6] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Massachusetts, USA, 1997.

[7] J.H. Hopcroft and J.D. Ullman, I*ntroduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1979.

[8] R. Rosenfeld, "A Maximum Entropy Approach to Adaptive Statistical Language Modeling," *Computer, Speech and Language*, vol. 10, pp. 187-228, 1996.

[9] A. Ganapathiraju, V. Goel, J. Picone, A. Corrada, G. Doddington, K. Kirchoff, M. Ordowski and B. Wheatley, "Syllable — A Promising Recognition Unit for LVCSR," *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 207-214, Santa Barbara, California, USA, December 1997.

[10] A. Ganapathiraju et. al., "WS97 Syllable Team Final Report," P*roceedings of the 1997 LVCSR Summer Research Workshop*, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, Maryland, USA, December 1997.

[11] D. Kahn, *Syllable-Based Generalizations in English Phonology*, Indiana University Linguistics Club, Bloomington, Indiana, USA, 1976.

[12] S. Greenberg, "Speaking in Shorthand - A Syllable-Centric Perspective for Understanding Pronunciation Variation," *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, Kerkrade, The Netherlands, May 3-6, 1998.

[13] K. F. Lee, "Context-dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, pp. 599-609, April 1990.

[14] K.F. Lee, *Large Vocabulary Speaker Independent Continuous Speech Recognition*, Ph. D. Thesis, Carnegie Mellon University, Pittsburgh, USA, 1988.

[15] H.A. Bourlard and N. Morgan, *Connectionist Speech Recognition — A Hybrid Approach*, Kluwer Academic Publishers, Boston, USA, 1994.

[16] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum Likelihood Estimation from Incomplete Data," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1-38, 1977.

[17] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley and Sons, New York, USA, 1991.

[18] E. Shriberg, "Disfluencies in SWITCHBOARD," P*roceedings of the International Conference on Spoken Language Processing*, Philadelphia, PA, USA, November 1996.

[19] E. Shriberg, and A. Stolcke, "How Far Do Speakers Back Up In Repairs? A Quantitative Model," *Proceedings of the International Conference on Spoken Language Processing*, Sydney, Australia, November 1998.

[20] J.L. Gauvain, et. al., "The LIMSI 1995 Hub3 System," *Proceedings of the DARPA Speech Recognition Workshop*, pp. 105-111, Harriman, NY, USA, February 1996.

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

[22] M.J. Russell and R.K., Moore, "Explicit Modeling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 5-8, Tampa, USA, 1985.

[23] L. E. Baum et. al., “A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Finite State Markov Model Chains,” *Annals of Mathematical Statistics*, vol. 41, pp. 164-171, 1970.

[24] V. Valtchev, *Discriminative Methods in HMM-based Speech Recognition*, Ph. D. Thesis, University of Cambridge, UK, 1995.

[25] B.-H, Juang and S. Katagiri, “Discriminative Training,” J*ournal of the Acoustical Society of Japan*, vol. 13, pp. 1404-1413, 1992.

[26] Y. Normandin, *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem*, Ph. D. Thesis, McGill University, Canada, 1991.

[27] P.C. Woodland and D.R. Cole, “Optimizing Hidden Markov Models using Discriminative Output Distributions,” *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April 1991.

[28] B.-H. Juang and S. Katagiri, “Discriminative Learning for Minimum Error Classification,” *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043-3054, 1992.

[29] W. Holmes, *Modelling Segmental Variability for Automatic Speech Recognition*, Ph. D. Thesis, University of London, UK, 1997.

[30] E. Osuna, et. al. “An Improved Training Algorithm for Support Vector Machines,” *Proceedings of the IEEE NNSP’97*, pp. 24-26, Amelia Island, USA, September 1997.

[31] G. Zoutendijk, *Methods in Feasible Directions — A Study in Linear and Non-linear Programming*, Elsevier Publishing Company, New York, NY, USA, 1960.

[32] B. Schölkopf, *Support Vector Learning*, Ph.D. Thesis, R. Oldenbourg Verlag Publications, Munich, Germany, 1997.

[33] T. Joachims, Making Large-scale SVM Learning Practical, In *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Cambridge, MA, USA., 1998.

[34] E.K.P. Chong and S.H. Zak, *An Introduction to Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 1996.

[35] O.L. Mangasarian, *Nonlinear Programming*, McGraw-Hill Book Company, New York, NY, USA, 1969.

[36] C. Cortes, V. Vapnik. Support Vector Networks, *Machine Learning*, vol. 20, pp. 293-297, 1995.

[37] C.J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, http://svm.research.bell-labs.com/SVMdoc.html, AT&T Bell Labs, November 1999.

[38] E. Osuna, R. Freund, and F. Girosi, "Support Vector Machines: Training and Applications," *MIT AI Memo 1602*, March, 1997.

[39] E. Osuna and F. Girosi, "Reducing the run-time complexity of Support Vector Machines", *Proceedings of the International Conference on Pattern Recognition*, Brisbane, Australia, 1998.

[40] M.A. Hearst, et. al., "Trends and Controversies - Support Vector Machines", *IEEE Intelligent Systems*, vol. 13, pp. 18-28, 1998.

[41] B. Schölkopf, C. Burges and A. Smola, *Advances in Kernel Methods: Support Vector Machines*, MIT Press, Cambridge, MA, USA, December 1998.

[42] T. Joachims, SVMLight: Support Vector Machine, http://www-ai.informatik.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/svm_light.eng.html, University of Dortmund, November 1999.

[43] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth International Group, 1984.

[44] T. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers", In *Advances in Neural Information Processing Systems 11*, MIT Press, Cambridge, MA, USA, 1998.

[45] M.D. Srinath, and P.K. Rajasekaran, *An Introduction to Statistical Signal Processing*, Wiley & Sons, New York, NY, USA, 1979.

[46] A. Ganapathiraju, J. Hamaker and J. Picone, "Support Vector Machines for Speech Recognition," *Proceedings of the International Conference on Spoken Language Processing*, pp. 2923-2926, Sydney, Australia, November 1998.

[47] A. J. Robinson, *Dynamic Error Propagation Networks*, Ph.D. Thesis, Cambridge University, UK, February 1989.

[48] G.D.Cook and A.J.Robinson, "The 1997 ABBOT System for the Transcription of Broadcast News," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, USA, 1998.

[49] N. Ström, “Phoneme Probability Estimation with Dynamic Sparsely Connected Artificial Neural Networks," *The Free Speech Journal*, vol. 1, no. 5, 1997.

[50] J. Tenenbaum, et. al., “Seperable Mixture Models,” In *Advances in Neural Information Processing Systems 9*, MIT Press, Cambridge, MA, USA., 1997.

[51] P. Ladefoged, *A Course in Phonetics*, Harcourt Brace Jovanovch, Inc., New York, NY, USA, 1975.

[52] R. Cole et al, “Alphadigit Corpus,” *http://www.cse.ogi.edu/CSLU/corpora/alphadigit*, Center for Spoken Language Understanding, Oregon Graduate Institute, 1997.

[53] J. Godfrey, E. Holliman, and J. McDaniel, “SWITCHBOARD: Telephone Speech Corpus for Research and Development,” P*roceedings of IEEE ICASSP*, San Francisco, California, USA, vol. 1, pp. 517-520, March 1992.

[54] J. Hamaker, et. al., “A Proposal for a Standard Partitioning of the OGI AlphaDigit Corpus,” available at http://www.isip.msstate.edu/resources/technology/projects/current/speech_recognition/research/syllable/alphadigits/

[55] N. Deshmukh, A. Ganapathiraju and J. Picone, “Hierarchical Search for Large Vocabulary Conversational Speech Recognition,” IEEE Signal Processing Magazine, vol. 1, no. 5, pp. 84-107, September 1999.

[56] S. Renals, *Speech and Neural Network Dynamics*, Ph. D. Thesis, University of Edinburgh, UK, 1990.

[57] M.D. Richard and R.P. Lippmann, “Neural Network Classifiers Estimate Bayesian a Posteriori Probabilities”, *Neural Computation*, vol. 3, no. 4, pp. 461-483, 1991.

[58] J. Tebelskis, S*peech Recognition using Neural Networks*, Ph. D. Thesis, Carnegie Mellon University, Pittsburg, PA, USA, 1995.

[59] J. Platt, Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, In *Advances in Large Margin Classifiers*, MIT Press, Cambridge, MA, USA, 1999.