Using Support Vector Machines for Speech Processing

Robert John Westwood

January 11, 1999

Abstract

This essay considers the topic "Explain how support vector machines work and discuss their potential advantages and disadvantages for use in speech pattern processing". Initially there is a discussion about the aims of statistical learning, and how this motivates the design of support vector machines. The advantages and disadvantages of using support vector machines for speech processing are discussed, incorporating a survey into the state of the art of this technology and possible research directions.

1 Introducing Support Vector Machines

1.1 Pattern Classification

The problem of pattern classification may be stated as follows. Suppose that we take as input an observation (strictly, a *statistic*, as we are unlikely to be able to capture all pertinent information). Suppose that we know that our observation can be categorised into exactly 1 of n disjoint *classes*, each having a probability distribution of class labels taken from l_1, ..., l_m. Normally the distribution will be a fixed label for each class, with $n = m$. The pattern classification problem is to estimate from which of the classes our observation was taken.

Pattern classification is said to be performed by a *classifier* or *classification machine*. Classifiers may be parameterised, so that the exact classification task which they perform may be altered. This means that, by adjusting the parameters, classifiers may undergo *statistical learning*. All classifiers considered in this essay, in particular the classifiers called *Support Vector Classifiers* or *Support Vector Machines* (SVMs) will be of this type.

With statistical learning techniques, we need some data with which to adjust the parameters (*train*) the classifier. This will consist of l observation, $\mathbf{x}_i$, $i = 1, ..., l$ and an associated class label y_i having values in $\{l_j\}$

indicating the true class which the observation was drawn from. This data will be referred to as the *training set*. In order to assess the effectiveness of the classifier, there will normally be a second set of data, the *test set*, which in the same way consists of observation - label pairs, but here the output of the classifier for a given observation will be compared with the true class label specified in the data. Thus an estimate for the accuracy of the classifier may be obtained.

1.2 Analysing Statistical Learning: Risk

Now let us formalise some of these concepts. Suppose that the observations and class labels are drawn according to some distribution function $P(\mathbf{x}, y)$. We will make the simplifying assumption that there are only 2 classes, and that the class labels are $+1$ and -1. Suppose that we have a machine which we wish to train to learn the mapping $\mathbf{x}_i \mapsto y_i$. The machine implements the function $f(\mathbf{x}, \boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is the vector of parameters, and so can be represented by this function. Note that we have assumed that the machine is deterministic. A particular choice of $\boldsymbol{\alpha}$ results in a *trained machine.*

The expectation of the test error over all possible observations is called the *risk*, *expected risk*, or *actual risk* [Burges, 1998], and can be expressed as

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

For a given training set we can compute the mean error rate, which we will call the *empirical risk*. Recalling that there are l observations in the training set, the definition of empirical risk, R_{emp} will be

$$R_{emp}(\boldsymbol{\alpha}) \quad = \quad \frac{1}{2l}\sum_{i=1}^{l} |y_i - f(\mathbf{x}_i, \boldsymbol{\alpha})| \tag{2}$$

where the quantity $\frac{1}{2}|y_i - f(x_i, a)|$ is called the loss. Under our assumptions the loss can only take the values 0 and 1.

In general, the actual risk may not determinable due to the requirement of knowing the distribution function P. It is possible to put a bound on $R(\boldsymbol{\alpha})$ [Vapnik & Cortes, 1995]. Take an η such that $0 \leq \eta < 1$. Then with probability $1 - \eta$, the following holds:

$$R(\boldsymbol{\alpha}) \quad \leq \quad R_{emp}(\boldsymbol{\alpha}) + \sqrt{\left(\frac{h(\log(\frac{2l}{h}) + 1) - \log(\frac{\eta}{4})}{l}\right)} \tag{3}$$

where h is a non-negative integer called the Vapnik Cervonekis (VC) dimension, and will be discussed later; the second term on the right-hand side of the equation is called the VC confidence. Notice that the right hand side does not depend on P, and so, if we know h we can compute the upper bound.

1.3 Minimising Risk

It is clear that we wish to minimise the expected error of classification over the possible observations, that is, we wish to minimise $R(\boldsymbol{\alpha})$. However, this is not generally possible, as we can not compute $R(\boldsymbol{\alpha})$ directly. We can however seek to minimise the bound on $R(\boldsymbol{\alpha})$, (3). In seeking to minimise the bound, we hope to minimise the quantity that we have bounded.

In many cases the principle of *empirical risk minimisation* (ERM) is employed [Gunn, 1998]. This uses R_{emp} as an estimate of R, and thus seeks to minimise the risk over the training set only. This technique may be successfully employed for many types of classification machines. One major difficulty of this technique is *overfitting*, or the lack of the ability to *generalise*. For most non-trivial classifiers, the training set can only be a (sparse) sample from the observation space. A classification machine may be over-trained on the training data, so that accuracy on this data is very high, but when presented with other data from the observation space, the misclassification error is also high. In order to try to assess the generalisation error, it is normal practice not to use a training and test sets with large intersection.

1.3.1 Machine Capacity

The ability of a machine to learn a training set is called the *capacity* of the machine. If the machine has insufficient capacity to learn the training set, then R_{emp} can not diminish to zero; if the machine has a great capacity, then the trained machine may be too closely modeled on the sample from the observation space used to train it, and so will not generalise well. Thus producing an optimum classifier should be a compromise between reducing the empirical error and reducing in the capacity of the machine.

One metric for the capacity of the machine is the VC dimension, which may be defined (in the case where we only have two classes, with labels $+1$ and -1), as follows. Suppose we have a set of functions, $S := \{f(\cdot, \boldsymbol{\alpha})\}$. Given l points in the observation space, these points can be labelled in 2^l different ways. If, for each labelling of these points, a function can be found in S such that it assigns the correct label to each point, then this set of points is said to be *shattered* by the set of functions S. Now, for the set of functions S, suppose that there is a positive integer h such that

1. there exists a set of h points from the observation space such that the points are shattered by S, and
2. for any k such that $k > h$, there does not exist a set of k points from the observation space that can be shattered by S.

Then h is said to be the Vapnik Chervonekis dimension. It can be seen that the VC dimension has some correspondence with the ability of a set of functions (or machines) to label (or learn) a training set.

1.3.2 Structural Risk Minimisation

An alternative approach to ERM is given by *structural risk minimisation*, which seeks to minimise both terms on the right-hand side of (3). Suppose that we have the same set S. We can form a sequence of subsets of S, $S_0 \subset S_1 \subset S_2 \subset \ldots$, and thus form a sequence of monotonically increasing positive integers, h_0, h_1, ..., where h_i is the VC dimension of S_i. Then for each subset, we can vary $\boldsymbol{\alpha}$ to produce a machine with minimal empirical risk. We can then take the machine whose sum of empirical risk and VC confidence is minimal.

It should be clear that this will minimise the bound on $R(\boldsymbol{\alpha})$, which will hopefully minimise $R(\boldsymbol{\alpha})$. More intuitively, we are choosing a machine that has sufficient capacity to learn the training set, but sufficiently small capacity to not overfit to the training set, and hence will generalise well to unseen data.

1.4 Linear Support Vector Machines For Linearly Separable Data

Support Vector Machines are one method which nearly embody the principle of structural risk minimisation (actually data-sensitive SRM [Bartlett & Shaw-Taylor, 1998]). In particular, it employs Occam's Razor [Vapnik & Cortes, 1995]: keep R_{emp} equal to zero, and minimise the VC confidence. We here consider the simplest case, that of data that is linearly separable into exactly two classes, labelled as -1 and $+1$. We also assume from here on that given any observation $\mathbf{x}$ there is exactly one class label y to assign to it. As the two classes are linearly separable, we can chose one of many oriented hyperplanes to separate the two classes. As the hyperplanes are oriented, given any observation we can determine which side of the hyperplane it is on, and thus assign the correct class label to the observation. If we were employing the ERM principle, then any of these hyperplanes would be equally good.

Intuitively, though, it is obvious that some hyperplanes are better than others, in terms of their ability to generalise to unseen data, as illustrated by figure 1. In particular, in the absence of information about the source distribution, one would expect a hyperplane that is further away from the training set data to generalise much better than one that comes close to the training set points. Define the *margin* to be the sum of the normal distances to the hyperplane of the point in each class that is closest to the hyperplane (figure 1). Ideally we would wish to show that minimising the margin minimises the VC dimension. This is not proven, but it is indicated by the result [Vapnik & Cortes, 1995; Smith, 1998] that minimising the margin minimises a bound on the VC dimension.

It may be noticed that only the observations closest in space to the separating hyperplane affect the choice of hyperplane. The other points may be moved around (providing that they do not come nearer to the hy-

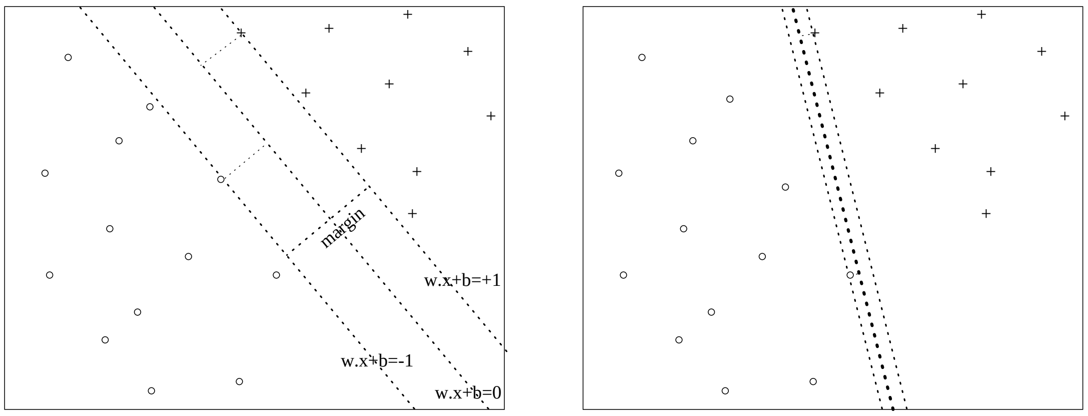

Figure 1: Figure showing separating hyperplanes with (left) maximal and (right) non-maximal margins

perplane than the closest points), or deleted, without affecting the choice of classifier (hyperplane). The vectors (observations) closest to the hyperplane are called *the support vectors*.

1.5 Non-Separable Case

In general, problems will not be linearly separable, and so it will be impossible for R_{emp} to be made equal to zero. Additionally, advantages may accrue from accepting some degree of misclassification error on the training set, in order to increase the ability of the network to generalise [Chin, 1998].

In the separable case, we first minimised R_{emp} and then maximised the margin. We can allow some degree of error by including a cost function of the number of errors; careful choice of this function will not (significantly) increase the complexitity of the optimisation. We will assume that the cost function is some suitable function, multiplied by a constant C, which we can vary to set the penalty for errors. A high value of C will reduce R_{emp}, but increases the likelihood of overfitting; low C will increase the generalisation ability of the machine but will increase R_{emp} [Smith, 1998].

Note that in this case many of the vectors in the training set may be deleted or moved within the class without changing the hyperplane chosen; those that may not be moved are still called support vectors, but the set will not just consist of those along the margin [Smith, 1998].

1.6 Training SVMs

It is outside of the scope of this essay to consider the mathematics behind SVM training in any detail; instead only the important points will be noted, for more details see [Vapnik & Cortes, 1995; Burgess, 1998].

The equation of the separating hyperplane will be given by

$$\mathbf{w}.\mathbf{x} + b = 0 \tag{4}$$

where $\mathbf{w}$ is the normal to the hyperplane. Notice that $(\mathbf{w}, b)$ are only specified up to a constant, so suppose that we chose $\mathbf{w}$ and b such that the training data satisfies the following constraints (the use of 1 as the constant is arbitrary, but convenient)

$$\mathbf{w}.\mathbf{x}_i + b \geq +1 \quad \text{for } y_i = +1 \tag{5}$$

$$\mathbf{w}.\mathbf{x}_i + b \leq -1 \quad \text{for } y_i = -1 \tag{6}$$

or equivalently

$$y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 \tag{7}$$

So we classify an observation vector $\mathbf{x}$ according to the sign of $\mathbf{w}.\mathbf{x} + b$, if the magnitude of the expression is less than unity, then the vector is in the margin. It can be shown [Burgess, 1998] that the size of the margin (which we wish to maximise) is $2/||\mathbf{w}||$, equivalently minimising $||\mathbf{w}||$, or $||\mathbf{w}||^2$. Hence, the minimisation task in the non-separable case is that of minimising this, plus the cost function associated with the errors.

The mathematical task of minimisation subject to constraints is performed by the technique of *Lagrange multipliers*. The formalisation considered so far is suitable for the application of the technique in the separable case, but requires some additional work in order to cope with the presence of errors.

An error is effectively a violation of the constraint (7). We do still wish these constraints to be held as much as possible (with this defined by the cost function). Hence we can introduce, in the standard way, positive *slack variables*, ξ_i, to allow the constraints to be weakened. We can rewrite the combined constraint (7) as follows

$$y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0 \quad \forall i \tag{8}$$

For a misclassification of vector $\mathbf{x}_i$ to occur, the corresponding slack variable ξ_i must have value greater than unity, hence $\Xi := \sum \xi_i$ is an upper bound on the number of errors in the training set, and so is suitable for use as the parameter for the cost function. The optimisation problem is kept sufficiently simple (is of the class [Burgess, 1998] of *convex* problems), for integer powers of Ξ, with the multiplicative constant C. For a cost function of $C\Xi$ or $C\Xi^2$, the problem reduces to that of the *quadratic* class.

Following the mathematics through [Vapnik & Cortes, 1995], it can be shown that the solution is a linear combination of the support vectors, and the support vectors are those such that $\xi_i \geq 1$, or $\xi_i = 0$ and (8) holds with equality [Smith, 1998].

1.7 Feature Space

Suppose that our support vectors are $\mathbf{s}_1, \ldots, \mathbf{s}_{n_s}$. As $\mathbf{w}$ can be expressed as a linear combination of these, the classification formula can be rewritten as follows

$$\Big(\sum_{i=1}^{n_s} \lambda_i y_i \mathbf{s}_i\Big).\mathbf{x} + b \tag{9}$$

As the dot product is a linear operator in its first term, we can rewrite the equation

$$\Big(\sum_{i=1}^{n_s} \lambda_i y_i \mathbf{s}_i.\mathbf{x}\Big) + b \tag{10}$$

Thus, the evaluation only depends on the dot product of the observed vector with each of the support vectors. It can also be shown [Burgess, 1998], that by solving the *dual* optimisation problem rather than the primal problem in the Lagrangian, that the vectors in the training set only appear in the presence of the dot product with another training set vector.

As training vectors and vectors to be classified only require a dot (*inner*) product operation, it is sufficient to require that they exist in a *Hilbert Space*, $\mathcal{H}$. This space need not be the input space, and is called the *feature space*. Let ϕ be a map from the observation space to the feature space, and the SVM will be a linear classifier in this feature space.

There is no constraint on ϕ to be linear. Hence, although the SVM will be implementing a linear classifier in the feature space, the corresponding classifier viewed in the input space will be non-linear. Also, there is no constraint that the dimensionality of the two spaces be the same, indeed, $\mathcal{H}$, may be infinite-dimensional.

It is beneficial to have the classifier work in a high-dimensional space for the following reasons.

1. It can be shown [Vapnik & Cortes, 1995] that the expected number of support vectors decreases as the dimension of the space increases. Thus the computation required for (10) becomes less.
2. More sophisticated classifiers can be implemented. For example, even for two dimensional data, a quadratic classifier requires 6 dimensions $(ax_0^2 + bx_0x_1 + cx_1^2 + dx_0 + ex_1 + f)$, for higher dimensional data or higher order classifiers the number of dimensions required in the feature space grows rapidly.

Working in a high-dimensional space normally is precluded on the grounds of computational and storage intensity. However, for an SVM, the only computation that we need to perform is $\langle\phi(\mathbf{x}^{(1)}), \phi(\mathbf{x}^{(2)})\rangle_{\mathcal{H}}$, for vectors in the observation space $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. Suppose that we can find a function K from the observation space to the reals, such that

$$K(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \equiv \langle\phi(\mathbf{x}^{(1)}, \phi(\mathbf{x}^{(2)})\rangle_{\mathcal{H}} \tag{11}$$

then, we can perform our computation in the input space, thus avoiding the complexity of working in $\mathcal{H}$. It is not even necessary to know $\mathcal{H}$, providing that we know that given some *kernel function* K there exists a ϕ and a $\mathcal{H}$. Sufficient conditions on K are given by *Mercer's condition* [Burgess, 1998]. Various kernels that are known to satisfy Mercer's condition are given in [Gunn, 1998].

1.8 Beyond Two Class Problems

The SVMs so far mentioned are binary classifiers only. Many classification problems are n-class problems. To be useful in these applications, the concepts need to be extended to be extended to deal with multiple classes. Different solutions are discussed in [Chin, 1998]; in summary the main options are

1. Use a one-to-rest classifier for each class (n classifiers in total)
2. Use one classifier for each pair of classes, with some form of voting scheme to combine the outputs of the $(n(n+1))/2$ classifiers
3. To extend the derivation of SVMs to the n-class problem.

As the comparison of these methods was performed using vowel recognition, further discussion will be postponed until a later section.

1.9 More on Training SVMs

Training an SVM involves minimising the Lagrangian dual problem. Hitherto, we have considered this from a mathematical perspective, and not considered the effects of finite numerical precision, nor performance considerations other than tractability.

The effect of numerical precision on the results of SVM training has been considered by [Chin, 1998], as well as the performance of different techniques for solving the Lagrangian. He concludes that the classification boundary of an SVM is very sensitive to the solution of the optimisation problem, and that there is no guarantee that a convergent solution converges to the optimum solution. However, on the evidence of the experiments documented, it is unclear that a sub-optimal solution is substantially worse than an optimum solution on data not in the training set.

The size of a training set is typically large. Even though the optimisation problem is tractable, typically $O(n^4)$ [Chin, 1998] on the number of training points, this can still result in a long training time. However, as stated above, the only vectors that are actually required are the support vectors, which typically are only a small proportion of the the total number of vectors in the training set. If these vectors can be determined early on in the training process, then the time taken will decrease rapidly. One such algorithm, which incrementally increases the amount of training data used, is described in [Chin, 1998], but results as to its effectiveness are mixed. In particularly, in does not appear to reduce the training time sufficiently to be of use for very large training sets. This paper also suggests that producing an optimiser specifically designed for the SVM optimisation problem may improve both the optimisation time and decrease the numerical precision problem.

2 SVMs in Speech Processing

Now that the motivation for support vector machines has been considered, and the theory behind SVMs has been discussed, we may move on to consider the relevance of SVMs to speech pattern classification. To give concreteness, we mainly consider the advantages and disadvantages for one possible problem in which SVMs may be deployed.

The archetypical classification problem in speech processing is that of speech recognition. Here we will consider one part of speech recognition, phone recognition, which is used in many (but not all) recognisers.

Incoming speech data is generally broken down into frames (which may overlap) and data reduction is performed on each frame. For certain subclasses of the phoneme recognition problem, the data may be reduced to a very small number of dimensions. For example, in the Barney-Peterson dataset only the frequencies of the first two formants are used for vowel recognition. The TIMIT dataset is also a corpus for vowel recognition systems, but here the data is reduced to 13 parameters (dimensions), 12 Mel-Frequency Cepstral Coefficients plus Energy (MFCC_E). More general systems will normally employ a higher number of parameters as the output to its data reduction, 39 being a common number.

All experiments using SVMs to classify speech data have so far only used one frame of data from the middle of the phone, when the effects of co-articulation are minimal.

2.1 Structure of Speech Processing Data

2.1.1 Temporality

The nature of speech data is to be temporal, wheras SVMs use static data. In the experiments based on phone recognition to date, then the data has already been segmented and only one frame of data used. As is pointed out by [Smith, 1998], a single frame of data is unlikely to be sufficient for recognition purposes, and it is suggested that three frames be used, at the beginning, middle and end of the phoneme. In addition, many speech recognition systems using MFCC_E data also incorporate first and even second differentials of each coefficient, provided some link with previous frames; data which has not currently been investigated as to whether it will increase the accuracy of SVMs.

Although these methods may provide the temporal links required for recognition of phones, they will not provide the demarcation of phone boundaries required. It is suggested in [Chin, 1998] that this may be performed by a HMM. As an SVM can be configured to act as other classifiers according to the choice of kernel function [Burgess, 1998], including two-layer neural networks, then the use of SVMs in this case would seem to be at least as good as using other classifiers. It is difficult to see how using an SVM with HMM phone marking would perform better than a HMM phone recogniser, particularly considering the problems of using a SVMs as n-class classifiers. Perhaps an approach taking advantages of the strengths of SVMs would be as a secondary classifier to a HMM phone recogniser, for example improving the recognition of minimal pairs like /b/ and /p/. However, research to date has concentrated on vowel recognition, and so the usefulness here has been untested.

An alternative to using a HMM to demark phone boundaries is to devise a recurrent structure as has been done for artificial neural networks, as suggested in [Smith, 1998].

2.1.2 Separability

It has been shown in [Smith, 1998] that if the input data is highly non-separable, then the accuracy of SVM classification is greatly diminished. It is not apparent how separable speech data is in general; in some cases it is highly non-separable, for example /UH/ *vs* /UW/ or /ER/ (ARPABET notation), in terms of the first and second formant frequencies. The suggestion has been made that using data from more frames, for example beginning, middle and end of the phone, will increase the separability of the data. If this is so, then the use of SVMs for classification of speech data will increase.

2.1.3 Errors

The formulation of SVMs was derived from the simplest classification problem: a binary classification of separable data. To circumvent the separable restriction, a cost function for errors was introduced. Although this did not much affect the formulation, the only motivation for choosing the form of the cost function was to keep the optimisation problem simple. In addition there is no motivation for choosing the value of C. It is intuitive that if the data is likely to contain many errors, the cost for misclassification should be low, whereas if the classes are intrinsically unseparable, than the cost should be high; but there is no theoretical justification for this. This is important because speech data will typically include lots of errors: for example, background noise, channel noise, channel differences, even day-to-day speech differences could be incorporated here.

2.2 Benefits of Dimensionality

In a typical phone recognition system the observation space will be high-dimensional, perhaps having 39 or more dimensions. Classifiers which are trained according to the empirical risk principle do not tend to generalise well on high-dimensional data, due to the sparseness of the training set in the space, unless the training set is very (perhaps impractically) large. SVMs are designed to generalise well regardless of the dimension of the input space. Indeed, the space that classification theoretically occurs in, the feature space, will typically be of a much higher dimension than the input space. Thus, one would expect SVMs to outperform similar classifiers on high dimensional data.

One experiment to empirically test this is described in [Smith, 1998]. The task was vowel recognition, using formant frequency or MFCC_E data using the Barney-Peterson and TIMIT data sets described above. An SVM classifier was implemented for both. A condensed nearest neighbour (CNN) classifier was also implemented for each problem. The CNN classifier was found superior on the two dimensional data, but on the thirteen dimensional TIMIT data, the SVM classifier was found to be superior.

2.3 Choice of Kernel

It would be thought that the choice of kernel is of critical importance in developing a SVM, as it will define the feature space. Most work on using SVMs in phone recognition has been based on using RBF kernels [Chin, 1998; Smith, 1998]. It is suggested in [Smith, 1998] that a polynomial kernel would be more appropriate, but it is concluded that for the TIMIT data employed, the choice of RBF or polynomial kernel made little difference to the accuracy of the classifier.

This does not present any evidence that the choice of kernel is unimportant though, it merely says that for this data, there is little to choose between RBF and polynomial kernels. Indeed, [Chin, 1998] shows how the choice or parameter for a kernel may have a large effect on the final classifier chosen, and so the choice of kernel may have as great or greater effect. It has already been stated that any knowledge as to the underlying statistical distribution of the process being observed is not made use when constructing an SVM. Indeed, that is presented as a strength of the SVM formulation, as it does not rely on knowledge of the underlying distribution function, and the method could thus be described as robust. However, if a good knowledge of the underlying distribution is known, then this knowledge will have been lost. In [Smith, 1998] there is an analytical method described for deriving a kernel function based on the distribution. On the Barney-Peterson data, taking the underlying distribution as a single or dual mixture of Normals, then test accuracy improves compared with an RBF kernel. It has already been seen from the comparison of CNN and SVM classifiers, that SVMs benefit from data in higher dimensions, but there are currently no results to indicate whether this improvement will generalise to higher dimensions. The disadvantage of this method is that the derivation is quite complex, and for many distributions it may not be possible to derive the related kernel. It is unclear as to whether a full derivation for many distributions could be performed. However, as much modelling of speech data is performed using mixtures of Normals, then the lack of distributions may not be such a problem as first thought.

Even once a kernel has been chosen the difficulties may not be over. Many kernels are parameterised, and one might wish to link the parameterisation of the kernel with training the SVM: it has already been mentioned how changing a kernel parameter can have a large effect on the chosen classifier. It has been suggested in [Gunn, 1998] that this may be performed for RBF kernels, but it is difficult to see how it may be performed in general.

Other ideas have been suggested as to how to incorporate knowledge of the problem into SVM classification [Burgess, 1998]. One such method is to create new ('virtual') data vectors based on support vectors [Schölkopf et al, 1996], and then to retrain the machine on both the original training set and the virtual vectors. The exact method depends on the problem. For example if phone classification was been performed on time domain data directly, then phone classification would be unchanged by a time shift, and so the virtual vectors would be created by time shifting the support vectors. More generally, the method could be used in speech to increase robustness across problems, for example to increase speaker-independence while decreasing the number of speakers required in the corpus.

2.4 Efficiency of Classification

Equation (10) gives the formula for determining a class for a linear 2-class SVM. Using (11), it can be seen that the equivalent equation for a non-linear SVM is

$$\Big(\sum_{i=1}^{n_s} \lambda_i y_i K(\mathbf{s}_i, \mathbf{x})\Big) + b \tag{12}$$

Thus the time it takes to compute the class of in input vector is mainly dependent on (i) the time to compute K, and (ii) the number of support vectors. The kernel K is within choice, and for most standard kernels the computation is not excessive [Gunn, 1998]. Thus, the major factor for determining the performance is the number of support vectors.

It is reported in [Smith, 1998] that on the Barney-Peterson data (2 dimensional) the SVM required approximately 100 support vectors, roughly a third of the training data. As stated in section 1.7, one expects the number of support vectors to go down as the dimension of the feature space increases. On the 13 dimensional TIMIT data set the maximum number of support vectors is still approximately 100 (124), but for certain experiments the number of support vectors falls to roughly 30 (28 for RBF kernel, 33 for polynomial). It is thus indicated that the number of support vectors does decrease with dimensionality; and also that performing a classification is an efficient operation.

2.5 Applying SVMs to Multiple Class Problems

Support Vector Machines are by their nature a binary classifier. In general, it is insufficient only to be able to recognise two phones. Section 1.8 discussed various methods by which the two class limit may be relaxed. The comparative accuracy of using one-to-rest and pairwise classifiers on TIMIT vowel data is discussed in [Chin, 1998]. Three different voting strategies were compared: a maximum votes cast algorithm (f_{max}), and two most probable class algorithms, using different probability estimates. As one may expect, the more sophisticated pairwise classifier was found to be generally the most effective. From the voting strategies though, the simplest one, (f_{max}) was the most accurate. However, performance wise, pairwise voting is quadratic in the number of classes, whereas one-to-rest classification is linear. As most phone recognition tasks involve much data and many classes, it seems doubtful that the benefits that may accrue from using pairwise classifiers would outweigh the performance handicap.

If one-to-rest classifiers are used however, then the nature of the training set will change. When only two classes are drawn from the training set, one would expect a roughly equal number of points in each class. When training a one-to-rest classifier, then the number of points in the 'one' class will be vastly smaller than the 'rest' class, especially for a large number of classes in the training set. This is important for two reasons; (i) it may affect generalisation performance, and (ii) the number of support vectors may be increased, which has time performance considerations, as discussed later. In [Smith, 1998], generalisation performance is measured by the accuracy of the classifier on a test set having trained the classifier under different conditions of data-inbalance. It was found that data-inbalance has a significant effect on the accuracy of the classifier (from 84% for a 50:50 split, to 63% for a 5:95 split), however the number of support vectors decreased markedly for inbalanced data, indicating that the larger class only needs to contribute a few points to form the simpler decision boundary with the now under-specified class.

It would seem then that neither pairwise or one-to-rest classifiers are particularly suitable for use in speech processing applications, the first for performance reasons, the second for accuracy considerations. There is though the possibility of a reformulation of SVM to the n-class problem. The accuracy and number of support vectors of two n-class classifiers (quadratic and linear) were compared against one-to-rest and pairwise classifiers on five data sets, including vowel data. The accuracy of the quadratic n-class classifier was commensurate with the other methods, whereas the linear classifier had, typically, an error rate twice of the other classifiers. However, the number of support vectors was much less for the n-class classifiers (and the linear much less than the quadratic), which would mean that during classification, the performance of the n-class classifier would be much greater than the methods for using a binary classifier. The training of n-class classifiers is likely to be more complicated though, although this has not yet been proven [Weston & Watkins, 1998].

2.6 Efficiency of Training

The training of SVMs has already been discussed in section 1.9. The corpora used in a training a typical phoneme recognition system will typically be large (but due to the dimensionality of the data, it will typically also be sparse). Thus both the difficulties of numerical precision and training time will affect the use of SVMs for this task, even with the decomposition method described in [Chin, 1998]. This would appear to be a large obstacle to the take up of using SVMs in speech processing.

It is also included in [Chin, 1998] that the training time is linear in the number of dimensions of the training data. This also would seem to indicate that SVMs are inappropriate for use in speech processing, as the dimensionality of the data is high. In fact, this would act against the use of SVMs in general, as one of their main advantages is the efficient computation and generalisability of classifiers, for high dimensional data. The exact trade-off between input space dimensionality and training time is obviously application specific, but there is currently very little data to determine exactly what the compromise is.

2.7 Speaker Identification

As an alternative application for speech processing, to further illustrate some of the points made above we consider the problem of speaker identification. An implementation for a a text-independent system for a fixed number of speakers is described in [Schmidt & Gish, 1996]. Instead of, as is conventional, identification using a Bayes decision rule based on statistical methods that involve estimating distribution functions, speakers are identified directly in the feature space. A polynomial kernel was employed of degrees 2, 3, 4 and 5. Both one-to-rest and pairwise classifiers were employed. In each case, the best results were obtained using the cubic polynomial. A slight improvement over a modified Gaussian system was observed. It is stated that the major advantage is that "computing polynomial classifiers from thousands of points is computational doable". It was also stated that it was expected that as the number of dimensions in the input space increase, the performance would improve. To use the higher dimensional space in other methods would be inappropriate, either due to the computational inefficiency, or the problems on training on a sparse data set.

3 Conclusions

The derivation of support vector machines was motivated by the desire to minimise the risk, $R(\boldsymbol{\alpha})$, a motivation that was grounded in the theory of statistical learning. However there is no guarantee that the SVM derivation actually minimises $R(\boldsymbol{\alpha})$. Having theoretical proof of this would prove that SVMs behave as claimed.

Despite this, there is empirical evidence that SVMs do offer good generalisation performance even in high dimensional space and with sparse training data. As SVMs may implement more traditional classifiers by careful choice of the kernel function, they should perform as least as well as these classifiers in terms of accuracy. As performance is dependent on the number of support vectors, performance is training data dependent; for sparse data this would appear to be relatively efficient.

There has been little work in the use of SVMs for speech processing. When SVMs have been implemented, reported test accuracies have been promising compared with similar classifiers. However, the time taken to train an SVM would seem to put it beyond practical application in speech processing for the time being.

Bibliography

[Bartlett & Shawe-Taylor, 1998] Bartlett, P. and Shawe-Taylor, J. (1998). Generalization Performance of Support Vector Machines and Other Pattern Classifiers. Learning Systems Group, Australian National University.

[Burgess, 1998] Burgess, C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. Kliwer Academic Publishers, Boston.

[Chin, 1998] Chin, K. (1998). Support Vector Machines applied to Speech Pattern Classification. Master's thesis, Department of Engineering, University of Cambridge.

[Gunn, 1998] Gunn, S. (1998). Support Vector Machines for Classification and Regression. Technical Report, Image Speech & Intelligent Systems Group, University of Southampton.

[Schmidt & Gish, 1996] Schmidt, M. and Gish, H. (1996). Speaker Identification via Support Vector Classifiers. *Proceedings, IEEE Conference on Acoustics, Speech and Signal Processing '96.*

[Schölkopf et al, 1996] Schölkopf, B., Burges, C. and Vapnik, V. (1996). Incorporating Invariances in Support Vector Learning Machines. *Springer Lecture Notes in Computer Science Vol. 1112, Berlin,47-5*

[Smith, 1998] Smith, N. (1998) Support Vector Machines applied to Speech Pattern Classification. Master's thesis, Department of Engineering, University of Cambridge.

[Vapnik & Cortes, 1995] Vapnik, V. and Cortes, C. (1995). Support-Vector Networks. *Machine Learning, Volume 20.*

[Weston & Watkins, 1998] Weston, J. and Watkins, C. (1998). Multi-class Support Vector Machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway (University of London).