FRAMEWISE PHONE CLASSIFICATION USING SUPPORT VECTOR MACHINES

Jesper Salomon

Department of Informatics
and Mathematical Modelling
Technical University of Denmark
c960404@gbar.dtu.dk

Simon King[1], *Miles Osborne*[2]

[1] Centre for Speech Technology Research
[2]Inst. for Communicating and Collaborating Systems
University of Edinburgh, UK
Simon.King@ed.ac.uk, osborne@cogsci.ed.ac.uk

ABSTRACT

We describe the use of Support Vector Machines for phonetic classification on the TIMIT corpus. Unlike previous work, in which entire phonemes are classified, our system operates in a *framewise* manner and is intended for use as the front-end of a hybrid system similar to ABBOT. We therefore avoid the problems of classifying variable-length vectors. Our frame-level phone classification accuracy on the complete TIMIT test set is competitive with other results from the literature. In addition, we address the serious problem of *scaling* Support Vector Machines by using the Kernel Fisher Discriminant.

1. INTRODUCTION

Most approaches to speech recognition involve learning some form of model from data. The model might be generative, as in the case of a Hidden Markov Model, or it might be a classifier, as in the case of a neural networks [1]. Typically, the model parameters are set to maximise the likelihood of the training data.

An alternative to learning models of the data is an example-based approach where a classifier is constructed in terms of actual training examples. One such approach is **Support Vector Machines** (SVMs) [2]. Whilst SVMs have been shown to yield competitive results in a number of domains (e.g. handwritten character recognition) they are not without problems. The problems addressed in this paper are: the choice of kernel and its parameters; building multiclass classifiers from inherently binary SVMs; the poor scaling of the standard training algorithm on large data sets; and the inability to directly interpret predicted outputs as probabilities. We give a brief overview of SVMs in section 1.1. Previous speech recognition work using Support Vector Machines (SVMs) [3, 4, 5, 6] has typically addressed the problem of classifying entire *segments* at once. The start and end times of these segments might be obtained from manual labelling (during training only), from a first pass using another model (e.g. a set of Hidden Markov Models), or may be simply hypothesised at recognition time as part of the search procedure. This approach has a serious theoretical problem: the classifier must deal with *variable length* feature vectors. Various ways round this problem have been tried, including resampling segments to a fixed length either by linear time warping [3] or *ad hoc* resampling [5], and building non-linear time-warping into the kernel [6].

We take a different approach, which is analogous to that in hybrid ASR systems such as ABBOT [1]. In these systems, a front end (e.g. a neural network or, in our case, a SVM) estimates a *posterior* phone conditional probability density function over the phone set on a frame-by-frame basis. This pdf is then decoded into a word sequence. Such an approach avoids the problem of dealing with variable-length feature vectors.

1.1. Support Vector Machines

Here we very briefly summarise the operation of Support Vector Machines, originally introduced by Vapnik [2]. SVMs are *kernel machines* and can be used for a variety of tasks, including that of pattern classification. In its basic form, a single SVM is a binary classifier which learns a decision boundary between two classes (e.g. two phonemes) in some *input space* (e.g. vectors of Mel-scale cepstral coefficients). We discuss approaches to multiclass problems (e.g. 1-of-40 phoneme classification for TIMIT) in section 1.2. To find a decision boundary between two classes a SVM attempts to *maximise the margin* between the classes, and choose linear separations in a *feature space*. A function called the *kernel* $K(\cdot)$ is used to project the data from *input space* to *feature space*, and if this projection is non-linear it allows non-linear decision boundaries. The formulation for SVMs is extremely simple. The classification of some known point in input space $\mathbf{x}_i$ is y_i, which is defined to be either -1 or $+1$. If $\mathbf{x}'$ is a point in input space with unknown classification, then

$$\hat{y}' = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}') + b\right) \quad (1)$$

where $y_i \in \{-1, 1\}$ and $\hat{y}'$ is the predicted class of point $\mathbf{x}'$. The function $K(\cdot)$ is the kernel, α_i are a set of adjustable weights and b is a bias, both to be learned during training. Classification time is linear in the number of support vectors. Various forms for the kernel function $K(\cdot)$ are possible, subject to certain constraints [2]. We investigated linear, polynomial and Gaussian kernels.

The training procedure for a SVM amounts to maximising the following expression using a set of N training vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ with corresponding known targets $\{y_1, y_2, \ldots, y_N\}$:

$$\sum_{i=1}^{N} \alpha_i \; - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad (2)$$

subject to $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$ where C is a user-defined parameter know as the *penalty term*. The set of support vectors in a trained SVM are those $\mathbf{x}_i$ for which $\alpha_i > 0$. It is clearly desirable for both storage and classification speed to have a *small* set of support vectors.

1.2. Multiclass SVMs

Various approaches are possible for constructing a multiclass classifier for a K-class problem from basic binary SVMs, e.g. [7]. The one we have found most successful is the one-versus-one scheme, where one classifier is trained for every possible pair of classes - resulting in $\frac{1}{2}K(K-1)$ classifiers ($K = 40$ in our experiments, so $\frac{1}{2}K(K-1) = 780$. Once a set of one-vs-one classifiers have been trained, there are two common ways of combining their outputs to construct a multiclass classifier:

One-vs-one voting scheme The first scheme was a simple majority vote: for a test frame all $\frac{1}{2}K(K-1)$ are run, each classifier casting one vote on favour of the class it chooses. The class with the most votes wins. Although this works reasonably well in practice (results are in section 2.3), there is clearly a flaw in such a scheme: the one-vs-one classifiers are forced to choose between only two classes, therefore many votes are *forced* to be cast for incorrect classes. In practice, these votes are distributed sufficiently uniformly as to not overrule votes from those binary classifiers who are choosing between a pair of classes which includes the correct class.

Directed acyclic graph (DAGSVM) scheme An alternative to the voting scheme is a greedy decision-graph-based algorithm based on [7]. In this scheme a *sequence* of binary classifications are performed using the same one-vs-one SVMs as above. If the first scheme was called "voting", this scheme might be called a "knockout competition". Now, only $(K-1)$ binary classifications are required to classify a test frame: $(K-1) = 39$ in our case.

2. EXPERIMENT 1: PHONETIC CLASSIFICATION

Maximising expression 2 using a general-purpose optimised quadratic programming algorithm will scale approximately as N^3 and require large amounts of memory. One approach to reducing training time on large data sets is to subdivide the training data into smaller chunks, effectively solving smaller optimisation tasks (i.e. ones with smaller N). Our experiments used the SVMTorch toolkit [8] which provides the sequential minimisation optimistation method. By using an extreme decomposition (dividing the problem into subsets of 2), this method effectively reduces the scaling factor to approximately N^2.

2.1. The data

We are currently working with the TIMIT corpus [9] because of it's high-quality phone labels. All results reported are framewise classification accuracies for the complete test set (the 1344 `si` and `sx` sentences which contain just over 50 thousand phones). The speech waveforms are parameterised in a standard way as Mel-scale cepstral coefficients (MFCCs) using 25ms frames spaced at 10ms intervals. Each input pattern $\mathbf{x}_i$ consists of the current frame of 12 MFCCs and energy plus delta and acceleration coefficients, and two context frames on each side, making a total of $(13 + 13 + 13) \times 5 = 195$ components. This formulation was arrived at by experimentation with varying numbers of context frames left and right of the frame being classified. The training set has about 1.1 million frames and the test set has about 400 thousand frames. Each frame has an associated 1-of-40 phonetic label derived from the TIMIT label files.

2.2. Choosing the kernel type and parameters

Kernel type An initial experiment was performed to determine the best kernel type. This used a reduced training set of 2000 frames per class for speed and we used the mean binary accuracy (i.e. the average across all 780 classifiers) on a validation set for comparisons. The Gaussian kernel performed best in all experiments, which is consistent with findings in [3] and [5], so all subsequent experiments use SVMs with Gaussian kernels.

Kernel parameters Depending on the kernel type, there are one or more parameters whose values are not learned but must be set by some other method. We investigated various methods for automatically determining the kernel parameters, but found that a simple exhaustive grid search of the parameter space produced the best values. This step is slow, and is therefore performed using only a subset of the training data.

We optimised the kernel parameters to maximise mean binary accuracy on a 4000-samples-per-class validation set.

For the Gaussian kernel, the parameters are the variance σ and the penalty term, C.

2.3. Results for the best system

The best system used Gaussian kernels and a total of 12.8% of the training frames were used as support vectors.

One-vs-one voting scheme *All* 780 classifiers must be evaluated to classify each unseen test frame using a simple majority voting scheme. The resulting accuracy is shown below:

system	framewise accuracy
one-vs-one	70.6%
DAGSVM	71.4%

DAGSVM system The DAGSVM scheme described earlier requires far fewer classifiers to classify a test frame (but the full set of 780 classifiers must be available, and therefore training is the same as for the voting scheme). By ordering the sequence of classifications to be initially between the most dissimilar phonemes (in terms of phonetic features) this system outperforms the voting scheme while reducing the classification time by a factor of $\frac{1}{2}K = 20$. The results are also shown in above table.

3. EXPERIMENT 2: THE SCALING PROBLEM

As mentioned earlier, the training algorithm implemented by SVMTorch scales approximately as N^2 for N training examples, so it is clearly going to be problematic for large data sets. In our previous experiments, we only managed to train using around 40% of the TIMIT training set (all test results are given on the *complete* test set however, so can be directly compared to other results from the literature). This took of the order of 10^4 (10 thousand) hours of CPU time on 750 MHz UltraSparc 3 processors[1]

We estimate 6 years of CPU time would be required for the full TIMIT training set. Even with ever increasing CPU speeds, training a SVM of the type in our first experiments is not going to be practical on larger data sets in the near future. **A solution to the scaling problem is the key to successfully using SVMs for speech recognition.**

In an attempt to solve the scaling problem, we chose to examine the Kernel Fisher Discriminant (KFD) introduced by Mika [10]. Like SVMs, it is another type of *kernel machine* that uses the "kernel trick" [2].

3.1. The Kernel Fisher Discriminant

The KFD is a version of Fisher's classical Linear Discriminant extended to incorporate kernels. The method attempts to find a decision boundary (in terms of support vectors)

[1]Since each binary SVM is trained independently, this step was performed in parallel.

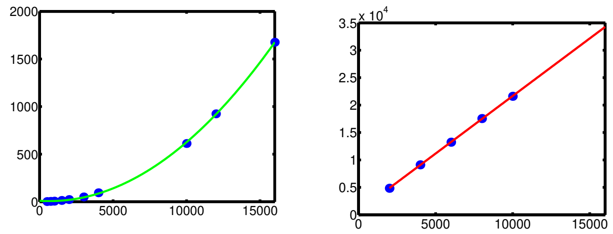

Fig. 1. Scaling of SVM (left) and KFD (right). Horizontal axis is training set size and vertical axis is training time in seconds.

that both maximises the distance between the class means and minimises the within-class variance. The mathematical formulation of the KFD can be written as a similar convex quadratic programming problem to SVMs:

$$\begin{array}{llcll} \text{Minimise} & \|\boldsymbol{\xi}\|^2 & + & C\cdot P(\boldsymbol{\alpha}) & \\ \text{Subject to} & \alpha_i K(\mathbf{x}_i,\mathbf{x}_j)+b & = & y_i+\xi_i & \forall i,j \\ \text{and} & \sum_{i\in\{y_i=+1\}} \xi_i = 0 & \text{and} & \sum_{i\in\{y_i=-1\}} \xi_i = 0 & \end{array} \quad (3)$$

The weights α_i, training patterns $\mathbf{x}_i$, targets y_i and bias b are the same as in the SVM formulation. The only difference is the regularising function, or prior, $P(\boldsymbol{\alpha})$. This function can be chosen freely, and is here set to $P(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|^2$. As with SVMs, classification of a test point is given by equation 1.

The KFD method has a very significant advantage over SVMs, which it inherits from Fisher's Discriminant: assuming Gaussian class distributions, **the outputs of a KFD can be interpreted as class-posterior probabilities**. This allows the model to be a direct replacement for the neural network in systems like [1].

3.2. A Sparse Greedy Approximation

The quadratic program in expression 3 can be solved efficiently by a Sparse Greedy Approximation technique [11]. This is an iterative technique that adds one support vector per iteration until the value of the expression reaches a predefined threshold. At each iteration, a predefined number of training patterns L are examined, and the one that best minimises the objective function is added to the set of support vectors.

The resulting *Sparse Greedy KFD* reduces scaling to $O(NM^2L)$ and memory requirements to $O(M^2)$, where N is the training set size, M the number of support vectors in the final solution, and L the number of training patterns examined in each iteration. If the required number of support vectors is low, this method completes training much faster than the SVM and with lower memory requirements, while still producing comparable performance (as seen below).

To compare the performance of the KFD to the SVM method, a selection of 10 binary one-vs-one problems were randomly chosen from the 780 required for the full problem. For each SVM, a 2000 frame training set was selected from the full training set with a separate validation set of 4000 frames; accuracy was measured on a 4000 frame test set. Kernel parameters were set as described in section 2.2. Training was stopped when a predefined maximum number of support vectors was reached[2]. The table below shows results comparing KFDs and SVMs on the same data sets. The KFD with a maximum of 200 support vectors per binary classifier produces almost the same performance as the SVM but uses only a third of the support vectors. Even with only 20 support vectors per KFD, performance remains surprisingly good. Figure 1 demonstrates that training time for KFDs scales linearly with training set size[3]

	Mean binary accuracy	Mean # SVs
SVM	91.75%	634
KFD(200)	91.50%	200
KFD(20)	88.45%	20

4. CONCLUSIONS

The framewise classification accuracy of the SVM is encouraging. Working with a subset of only 40% of the full TIMIT training set, it produced results comparable to the best results found in the literature [12, 1]. By using Mika's Sparse Greedy KFD, we demonstrate a solution to two of the main problems of SVMs: scaling, and creating sparse and *fast* classifiers. However, if a large number of support vectors M are required, even the KFD is slow, since it scales with a factor M^2.

Future work We are currently running experiments on the *full* TIMIT training set using the Sparse Greedy KFD method. Since the outputs of the KFD can be interpreted as posterior class probabilities, we intend to decode the framewise output from our KFD model in a similar fashion to [1]. Our ultimate goal is to move on to larger data sets.

4.1. Acknowledgments

Ronan Collobert (Université de Montréal, Canada and IDIAP, Switzerland) provided the SVMTorch software. We are deeply grateful to Sebastian Mika (GMD First, Germany) for software and advice for the KFD method. Thanks also to Lars Kai Hansen (DTU, Denmark).

[2] Since the maximum number of support vectors can be defined by the user, this parameter can be adjusted to suit the difficulty of the task.

[3] Absolute values for training time for the two methods cannot be compared because they were implemented independently and were run on differing hardware.

5. REFERENCES

[1] Tony Robinson, G.D. Cook, D.P.W. Ellis, E. Fosler-Lussier, S.J. Renals, and D.A.G. Williams, "Connectionist speech recognition of broadcast news," *Speech Communication*, 2001.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, N.Y., 1995.

[3] P. Clarkson and P.J Moreno, "On the use of support vector machines for phonetic classification," in *Acoustics, Speech and Signal Processing, volume II*, 2000, pp. 585–588.

[4] N. Smith and M. Niranjan, "Data-dependent kernels in SVM classification of speech patterns," in *Proc. ICSLP*, Beijing, 2000, pp. 297–300.

[5] A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid SVM/HMM architectures for speech recognition," in *Proc. ICSLP*, Beijing, 2000.

[6] Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, and Shigeki Sagayama, "Support vector machine with dynamic time-alignment kernel for speech recognition," in *Proc. Eurospeech*, September 2001.

[7] N. Christianini J. Platt and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," Tech. Rep., Microsoft Research, Redmond, US, 1999.

[8] Ronan Collobert and Samy Bengio, "SVMTorch: Support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[9] J. S. Garofolo, *Getting started with the DARPA TIMIT CD-ROM: An acoustic phonetic continuous speech database*, National Institute of Standards and Technology (NIST), Gaithersburgh, MD, 1988.

[10] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. 1999, pp. 41–48, IEEE.

[11] S. Mika, A. Smola, and B. Scholkopf, "An improved training algorithm for kernel fisher discriminants," 2001.

[12] Ruxin Chen and L. H. Jamieson, "Experiments on the implementation of recurrent neural networks for speech phone recognition," in *Proc. of the Thirtieth Annual Asilomar Conference on Signals, Systems and Computers*, November 1996.