Support Vector Machines for Noise Robust ASR

M. J. F. Gales, A. Ragni, H. AlDamarki and C. Gautier

Cambridge University Engineering Department
Trumpington St., Cambridge CB2 1PZ, U.K.
{mjfg,ar527,hia21,cg291}@eng.cam.ac.uk

***Abstract*—Using discriminative classifiers, such as Support Vector Machines (SVMs) in combination with, or as an alternative to, Hidden Markov Models (HMMs) has a number of advantages for difficult speech recognition tasks. For example, the models can make use of additional dependencies in the observation sequences than HMMs provided the appropriate form of kernel is used. However standard SVMs are binary classifiers, and speech is a multi-class problem. Furthermore, to train SVMs to distinguish word pairs requires that each word appears in the training data. This paper examines both of these limitations. Tree-based reduction approaches for multi-class classification are described, as well as some of the issues in applying them to dynamic data, such as speech. To address the training data issues, a simplified version of HMM-based synthesis can be used, which allows data for any word-pair to be generated. These approaches are evaluated on two noise corrupted digit sequence tasks: AURORA 2.0; and actual in-car collected data.**

I. Introduction

For difficult speech classification tasks, such as classifying speech in low Signal-to-Noise Ratio (SNR) conditions, standard approaches based on Hidden Markov Models (HMMs) may not achieve acceptable levels of performance. To address this problem schemes such as *acoustic code-breaking* [1] may be used. Here a second classifier, for example based on Support Vector Machines (SVMs) [2], is used to resolve highly confusable data that the standard recogniser is not able to handle. One of the issues with using kernel-based classifiers, such as SVMs, in varying noise conditions is that the classifiers need to be adapted to the test data acoustic condition. The simplest approach to do this is to adapt the kernel to the noise condition [3]. For generative kernels [4], [5], standard model-based noise robustness schemes such as Vector Taylor Series (VTS) [6] can be used. Though performance gains were obtained over standard VTS compensation [3] on a continuous digit recognition task, there are still a number of issues that need to be addressed to allow this form of classification to be more generally applied. This paper describes initial work on two of these problems: efficient multi-class classification; and training SVMs where no examples of the word are available in the training data.

The standard SVM implementation is a binary classifier [2]. To modify SVMs to handle multi-class problems there are two basic approaches. SVM training and classification can be modified to directly support multi-class problems [7], [8], [9]. The issue with this approach is that the training algorithm is made more complicated and scales poorly as the number of training examples and classes increases. When using generative kernels [4], [5], which allow the time varying nature of speech to be handled, there is an additional problem. The score-spaces associated with generative kernels are determined by the generative models of the classes to be classified. Thus for a multi-class SVM, where the same score-space must be used for all classes, either a very large composite score-space derived from all class generatives must be used, or a single global generative must be used, effectively a *Fisher kernel* [10]. The alternative approach, and the one examined in this paper is a reduction style scheme using tree-based classifiers [11], [12], [13]. This requires no changes to the training, and classifiers may be trained in parallel. This paper describes three forms of tree-based classifier: baseline Directed Acyclic Graphs (DAGs); Filter-Trees; and Adaptive DAG (ADAG) approaches. These schemes differ in terms of the style/number of classifiers and decoding cost.

The second issue addressed in this paper is how to train word-pair classifiers where there is limited, or no, examples of the words in the training data. This is a known limitation of schemes such as acoustic code-breaking [1] and means the schemes cannot be applied to tasks such as city-name recognition. One option to address this would be to alter the level at which the classifiers operate, for example training phone-based classifiers. However this dramatically complicates the issue of how to specify the phone-boundaries, required for these forms of classifier. The approach adopted in this work is to artificially generate training data. This is effectively a simplified version of speech synthesis where only the parameterised speech data is required, not the waveform. Recently HMM Statistical Speech Synthesis (HTS) [14] has become increasingly popular. These model-based approaches are suited for the task in-hand as the models themselves can be compensated for a particular noise condition and used to generate "noise-corrupted" speech data. This is not possible with concatenative approaches (however noise can be directly added to the waveform).

II. Adapting SVMs to Noise

Support Vector Machines (SVMs) [2] are an approximate implementation of structural risk minimisation. They have been found to yield good performance on a wide range of tasks. The theory behind SVMs has been extensively described in many papers and is not discussed here. This section concentrates on how SVMs can be applied to tasks where there is sequence data, for example speech recognition.

One of the issues with applying SVMs to sequence data, such as speech, is that the SVM is inherently static; "obser-

vations" (or sequences) are all required to be of the same dimension. A range of *dynamic kernels* have been proposed that handle this problem. Of particular interest in this work are those kernels that are based on generative models [4], [5]. In these approaches a generative model is used to determine the feature-space for the kernel. An example first-order feature-space for a generative kernel with observation sequence $\mathbf{Y}$ may be written as

$$\boldsymbol{\phi}(\mathbf{Y};\boldsymbol{\lambda}) = \frac{1}{T}\begin{bmatrix} \log\left(\frac{p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_1)})}{p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_2)})}\right) \\ \boldsymbol{\nabla}_{\lambda^{(\omega_1)}} \log p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_1)}) \\ \boldsymbol{\nabla}_{\lambda^{(\omega_2)}} \log p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_2)}) \end{bmatrix} \quad (1)$$

where $p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_1)})$ and $p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_2)})$ are the likelihood of the data using generative models associated with classes ω_1 and ω_2 respectively. HMMs are used as the generative model in this paper and only the derivatives with respect to the means are used, though it is possible to use other, and higher-order, derivatives. As SVM training is a distance based learning scheme it is necessary to define an appropriate metric for the distance between two points. In common with other work in this area [4], [5], the metric $\mathbf{G}$ is set to the diagonalised empirical covariance matrix of the training data.

Classification using this form of generative kernel with observation sequence $\mathbf{Y}$ and training data $\mathbf{Y}_1,\ldots,\mathbf{Y}_n$ is then based on the SVM score $\mathcal{S}(\mathbf{Y})$

$$\mathcal{S}(\mathbf{Y}) = \sum_{i=1}^{n} \alpha_i^{\mathtt{svm}} z_i K(\mathbf{Y}_i,\mathbf{Y};\boldsymbol{\lambda}) + b \quad (2)$$

where $\alpha_i^{\mathtt{svm}}$ is the Lagrange multiplier for observation sequence $\mathbf{Y}_i$ obtained from the SVM maximum margin training and b is the bias (these are trained for each word-pair). $z_i \in \{1,-1\}$ indicates whether the sequence was a positive (ω_1) or negative (ω_2) example, and $K(\mathbf{Y}_i,\mathbf{Y}_j;\boldsymbol{\lambda}) = \boldsymbol{\phi}(\mathbf{Y}_i;\boldsymbol{\lambda})^{\mathsf{T}}\mathbf{G}^{-1}\boldsymbol{\phi}(\mathbf{Y}_j;\boldsymbol{\lambda})$.

To adapt the SVM classifiers it is necessary to modify the SVM classification rule. There are two options. The Lagrange multipliers, $\{\alpha_1^{\mathtt{svm}},\ldots,\alpha_n^{\mathtt{svm}}\}$, may be modified. However with very limited data in the target domain, in these experiments a single utterance, this is not possible. Here the parameters associated with the generative kernel $\boldsymbol{\lambda}$ are modified instead [3]. This can be achieved using any model-based compensation scheme. The Lagrange multipliers are then noise-independent.

VTS model-based compensation is a popular approach for model-based compensation [15], [16], [6]. There are a number of possible forms that have been examined. In this work the first-order VTS scheme described in [16] is used. A brief summary of the scheme is given here. The static *mismatch function*, mean, $\boldsymbol{\mu}_{\mathsf{y}}^{\mathsf{s}}$, and covariance matrix, $\boldsymbol{\Sigma}_{\mathsf{y}}^{\mathsf{s}}$, of the corrupted speech distribution are given by [6]

$$\begin{aligned} \boldsymbol{y}_t^{\mathsf{s}} &= \boldsymbol{x}_t^{\mathsf{s}} + \boldsymbol{h} + \mathbf{C}\log\left(\mathbf{1}+\exp(\mathbf{C}^{-1}(\boldsymbol{n}_t^{\mathsf{s}} - \boldsymbol{x}_t^{\mathsf{s}} - \boldsymbol{h}))\right) \\ &= \boldsymbol{x}_t^{\mathsf{s}} + \boldsymbol{h} + \boldsymbol{f}(\boldsymbol{n}_t^{\mathsf{s}} - \boldsymbol{x}_t^{\mathsf{s}} - \boldsymbol{h}) \quad (3) \\ \boldsymbol{\mu}_{\mathsf{y}}^{\mathsf{s}} &= \boldsymbol{\mu}_{\mathsf{x}}^{\mathsf{s}} + \boldsymbol{\mu}_{\mathsf{h}} + \boldsymbol{f}(\boldsymbol{\mu}_{\mathsf{n}}^{\mathsf{s}} - \boldsymbol{\mu}_{\mathsf{x}}^{\mathsf{s}} - \boldsymbol{\mu}_{\mathsf{h}}) \quad (4) \\ \boldsymbol{\Sigma}_{\mathsf{y}}^{\mathsf{s}} &= \mathrm{diag}\left(\mathbf{A}\boldsymbol{\Sigma}_{\mathsf{x}}^{\mathsf{s}}\mathbf{A}^{\mathsf{T}} + (\mathbf{I}-\mathbf{A})\boldsymbol{\Sigma}_{\mathsf{n}}^{\mathsf{s}}(\mathbf{I}-\mathbf{A})^{\mathsf{T}}\right) \quad (5) \end{aligned}$$

where matrix $\mathbf{A}$ above is the partial derivative, $\partial \boldsymbol{y}^{\mathsf{s}}/\partial \boldsymbol{x}^{\mathsf{s}}$, evaluated at $\overline{\boldsymbol{\mu}}^{\mathsf{s}} = \boldsymbol{\mu}_{\mathsf{n}}^{\mathsf{s}} - \boldsymbol{\mu}_{\mathsf{x}}^{\mathsf{s}} - \boldsymbol{\mu}_{\mathsf{h}}$. This yields

$$\mathbf{A} = \partial \boldsymbol{y}^{\mathsf{s}}/\partial \boldsymbol{x}^{\mathsf{s}} = \mathbf{C}\mathbf{F}\mathbf{C}^{-1} \quad (6)$$

where $\mathbf{F}$ is a diagonal matrix with elements given by $\mathbf{1}/(\mathbf{1}+\exp(2\mathbf{C}^{-1}(\overline{\boldsymbol{\mu}}^{\mathsf{s}}))$, and $\mathbf{C}$ is the DCT matrix. Similar expressions can be found for the dynamic parameter compensation using the *continuous time approximation*. The VTS compensated parameters will be referred to as $\boldsymbol{\lambda}_{\mathsf{y}}$.

The compensation schemes described above have assumed that the noise model parameters, $\boldsymbol{\mu}_{\mathsf{n}}$, $\boldsymbol{\Sigma}_{\mathsf{n}}$ and $\boldsymbol{\mu}_{\mathsf{h}}$, are known. In practice these are seldom known in advance so must be estimated from the test data. In this work the noise estimation is based on the Maximum Likelihood (ML) noise estimation scheme described in [16]. In addition, the Hessian approach for the noise variance in [15] was implemented. This has no effect on recognition performance, but improves the estimation speed as there are fewer back-offs to ensure that the auxiliary function increases.

III. Multi-Class SVM Classification

There are a number of options to extend the standard binary SVM classifier to handle multi-class problems. The first category is to modify the SVM training and classification to handle multi-classes [7], [8], [9]. Though various options for this exist, they involve additional complexity in the training algorithm. This paper only considers the second option which is to use *reduction* style approaches to reduce multi-class classification to a set of binary classification problems. Two forms of classifier will be examined: multiple 1-v-1 classifiers; and tree-based classifiers.

For all the forms of classifier investigated the following initial stages are run to segment the continuous data:

1) Compensate the acoustic models for the test condition
2) Recognise the test utterance $\mathbf{Y}$ to obtain 1-best hypothesis, $\mathbf{h} = h_1,\ldots,h_K$ and align to give the word-segmented data sequence $\tilde{\mathbf{Y}}_1,\ldots,\tilde{\mathbf{Y}}_K$

The task is then to classify each of the word segments $\tilde{\mathbf{Y}}_i$.

A. 1-v-1 Classifiers

There are a range of options available for using simple voting schemes with SVMs for multi-class classification. The simplest is to use a one-versus-the rest classifier[1]. The alternative, and the baseline approach adopted here, is the one-versus-one (1-v-1). The application of 1-v-1 voting for continuous speech task uses the following process during recognition:

1) For each segment $\tilde{\mathbf{Y}}_i$:
a) for each word pair $\{\omega_l,\omega_j\}$ set $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_{\mathsf{y}}^{(\omega_l)},\boldsymbol{\lambda}_{\mathsf{y}}^{(\omega_j)}\}$

$$\hat{\omega} = \begin{cases} \omega_l, & \text{if } \mathcal{S}(\tilde{\mathbf{Y}}_i) + \epsilon \log\left(\frac{p(\tilde{\mathbf{Y}}_i;\boldsymbol{\lambda}_{\mathsf{y}}^{(\omega_l)})}{p(\tilde{\mathbf{Y}}_i;\boldsymbol{\lambda}_{\mathsf{y}}^{(\omega_j)})}\right) \geq 0 \\ \omega_j; & \text{otherwise} \end{cases} \quad (7)$$

count $[\hat{\omega}]$ = count $[\hat{\omega}]$ + 1

[1]This form of classifier was not investigated as the grouping of classes for speech recognition was not found to yield good results [17].

b) classification, $\hat{h}_i$, is given by:
1) if no ties in voting: $\hat{h}_i = \text{argmax}_\omega \{\text{count}[\omega]\}$
2) if only two words (ω_l, ω_j) tie then $\hat{h}_i$ determined using the result from that pair in equation 7
3) if more than two words tie $\hat{h}_i = h_i$

ϵ is an empirically set scaling value, as the log-likelihood ratio is expected to be the most discriminatory dimension.

As an alternative approach which does not require SVMs to be trained for all possible pairs is a *cascade* approach. Here a subset of SVMs are applied in order to the segmented data. Thus the procedure is:

1) For each segment $\tilde{\mathbf{Y}}_i$, initialise $\hat{\omega} = h_i$:
a) for each word pair $\{\omega_l, \omega_j\}$ set $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_{\text{y}}^{(\omega_l)}, \boldsymbol{\lambda}_{\text{y}}^{(\omega_j)}\}$: if $\hat{\omega} = \omega_l$ or $\hat{\omega} = \omega_j$ apply classification rule (7)
b) classification $\hat{h}_i = \hat{\omega}$

This cascade approach enables a subset of all word-pair classifiers to be trained, allowing the number of classifiers (and decoding operations) to be determined by the designer.

B. Tree-Based Classifiers

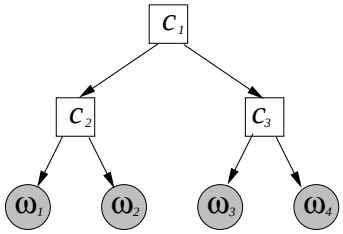

Fig. 1. Example tree-based classifier structure

Tree-based reduction techniques are common in the general pattern processing literature [18], [11], [12], [13]. The basic process is to convert the multi-class classification process into a sequence of binary classification processes. A binary tree (used in this work) for a 4-class problem is shown in figure 1. There are a number of options for both training and classifying with these forms of tree-based approaches. An interesting aspect when applying these approaches with generative kernels, which differs from the previous applications of these tree-based approaches, is that the score-spaces associated with the tree are functions of the classes being classified (the HMMs change). This allows additional flexibility.

For the bottom layer of classifiers in figure 1, c_2 and c_3, the training and classification does not depend on the form of tree training and classification. The differences between the schemes is in the form of training and classification with c_1. For this work three forms of classifier were investigated.

Divide-By-Two DAG (DAG): here the classifier c_1 is trained to classify $\{\omega_1, \omega_2\}$-v-$\{\omega_3, \omega_4\}$ [18]. This classifier uses all the training data associated with these classes to train the SVM. There is a choice in the form of the score-space that can be used. The simplest option is to train composite HMMs for each of the sets $\{\omega_1, \omega_2\}$ and $\{\omega_3, \omega_4\}$ and then construct the space as usual. However this was found to yield poor performance (see [17] for details). The alternative is to use the score-spaces derived from the individual class HMMs and select a subset of these for classification. This is the approach adopted here. In the same fashion, the log-likelihood ratio term, the first score-space element has flexibility, needs to be defined. Here the following form is used

$$\phi_1(\tilde{\mathbf{Y}}; \boldsymbol{\lambda}) = \frac{1}{T} \log \left(\frac{\max \left\{ p(\tilde{\mathbf{Y}}; \boldsymbol{\lambda}_{\text{y}}^{(\omega_1)}), p(\tilde{\mathbf{Y}}; \boldsymbol{\lambda}_{\text{y}}^{(\omega_2)}) \right\}}{\max \left\{ p(\tilde{\mathbf{Y}}; \boldsymbol{\lambda}_{\text{y}}^{(\omega_3)}), p(\tilde{\mathbf{Y}}; \boldsymbol{\lambda}_{\text{y}}^{(\omega_4)}) \right\}} \right) \quad (8)$$

By selecting the first element in this fashion as $\epsilon \rightarrow \infty$ the performance of the tree-based classifier is the same as that of the HMM-based classification scheme. Classification with this form of tree is performed by starting at the first classifier c_1 and proceed down the tree following each decision.

Filter-Tree (FT): if lower level classifiers in the tree do not correctly classify the data, then it is irrelevant whether the higher-level classifiers classify the data correctly or not. This is the basic concept behind filter-trees [11] [2]. Training proceeds from the bottom up. Only data that is correctly classified by the lower level classifiers, c_2 and c_3, is used for training the higher level classifier, c_1. This allows the higher level classifiers to concentrate on data that can be correctly classified. The same score-spaces as the DAG scheme above are used. Classification is top-down in the same fashion as the DAG approach.

Adaptive DAG (ADAG): for the top-down classification schemes most of the mistakes result from errors in the higher-level classifiers, e.g. c_1 [17]. To address this problem the ADAG approach [12] can be used. This follows the rules of a tennis tournament. Classification proceeds from the bottom-up. Thus c_2 and c_3 are used to initially classify the data. The results from these classification tasks, for example ω_1 and ω_4, are passed to the higher level classifier. Rather than having a fixed classifier at this stage, i.e. c_1, the classifier changes according to the lower level results. Thus in this example c_1 would use the classifier ω_1-v-ω_4. It is also possible to train the higher-level classifiers using the filter-tree approach above [11]. This is then referred to as filter-tree decoding. However for the tasks considered here there were very few training data classification errors, so the performance of the filter-tree decoding was the same as the ADAG approach.

TABLE I
NUMBER OF CLASSIFIERS AND DECODES FOR A K-CLASS TASK

Scheme	# Classifiers	# Decode
1-v-1	$K(K-1)/2$	$K(K-1)/2$
DAG	$K-1$	$\log_2(K)$
FT	$K-1$	$\log_2(K)$
ADAG	$K(K-1)/2$	$K-1$

For the three tree-based approaches discussed above there are differences in the number of classifiers that need to be trained, as well as the number of classifications to get the final result. Table I contrasts the approaches for a K-class classification problem. For the smallest number of classifiers, and smallest number of classifications, the top-down DAG and

[2]The general filter-tree training process is more general allowing costs for misclassification to be incorporated in the process. For the simple uniform cost scheme considered here this is unnecessary.

FT approaches should be used. 1-v-1 and ADAG have the same number of classifiers, but differ in the decode cost. All these approaches scale as K increases. The cascade approach, described earlier, does not.

IV. Extended Acoustic Code-Breaking

For the SVM schemes described above it is necessary to have sufficient acoustic training data for each of the word-pair SVMs to be trained. Though this is possible for some tasks, for others, such as city name recognition, it is unlikely that there will be sufficient data. One option to address this would be to use SVM phone classifiers. However this is complicated as the phone boundaries will be significantly harder to estimate than the word boundaries, and even more sensitive to the precise phone sequence being considered. Alternatively data can be artificially generated. This is the approach adopted in this work. Effectively this is a restricted form of speech synthesis. Rather than needing to generate waveforms, only the parameterised speech sequences need to be generated. Thus many of the issues associated with speech synthesis, such as excitation and prosody, are not relevant to this task. Currently there are two main forms of speech synthesis: concatenative; and HMM-based. For this work as there will be interest in adapting to a particular noise-condition the parametric form of HMM-based synthesis will be examined.

A. HMM Synthesis

The simplest approach to synthesis with HMMs is to directly use them to generate data. Samples are drawn from

$$p(\mathbf{Y}) = \sum_{\boldsymbol{\omega},\mathbf{q}} P(\boldsymbol{\omega})P(\mathbf{q}|\boldsymbol{\omega})\mathcal{N}(\mathbf{Y};\boldsymbol{\mu}_{\mathbf{q}},\boldsymbol{\Sigma}_{\mathbf{q}}) \tag{9}$$

where $\mathbf{Y}$ is the complete sequence of static, delta and delta-delta MFCCs, and the mean, $\boldsymbol{\mu}_{\mathbf{q}}$, and covariances, $\boldsymbol{\Sigma}_{\mathbf{q}}$, are

$$\boldsymbol{\mu}_{\mathbf{q}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathsf{y}}^{(q_1)} \\ \vdots \\ \boldsymbol{\mu}_{\mathsf{y}}^{(q_T)} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{\mathbf{q}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathsf{y}}^{(q_1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\Sigma}_{\mathsf{y}}^{(q_T)} \end{bmatrix} \tag{10}$$

$\mathbf{q}$ is the state/component sequence, and $\boldsymbol{\omega}$ the word sequence. This is a simple generative process, but the generated observations will be based on the same conditionally independence assumptions as the underlying HMMs.

B. HMM Statistical Speech Synthesis

To overcome the conditional independence assumptions that are often cited as an issue with the standard HMM synthesis, HTS-based synthesis can be used [14]. Here acoustic models with static and dynamic parameters are used to obtain a distribution for the underlying static sequence. Consider the sequence (static observations with simple differences)

$$\begin{bmatrix} \boldsymbol{y}_{t-1} \\ \boldsymbol{y}_t \\ \boldsymbol{y}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\frac{\mathbf{I}}{2} & \mathbf{0} & \frac{\mathbf{I}}{2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{\mathbf{I}}{2} & \mathbf{0} & \frac{\mathbf{I}}{2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{\mathbf{I}}{2} & \mathbf{0} & \frac{\mathbf{I}}{2} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}^{\mathsf{s}}_{t-2} \\ \boldsymbol{y}^{\mathsf{s}}_{t-1} \\ \boldsymbol{y}^{\mathsf{s}}_{t} \\ \boldsymbol{y}^{\mathsf{s}}_{t+1} \\ \boldsymbol{y}^{\mathsf{s}}_{t+2} \end{bmatrix} \tag{11}$$

It is then possible to express the observed sequence $\mathbf{Y}$ in terms of the underlying static sequence by $\mathbf{Y} = \mathbf{A}\mathbf{Y}^{\mathsf{s}}$. The likelihood of a static sequence can be expressed in terms of the HMM features as

$$\frac{1}{Z}p(\mathbf{A}\mathbf{Y}^{\mathsf{s}}|\mathbf{q};\boldsymbol{\lambda}) = \frac{1}{Z}\mathcal{N}(\mathbf{Y}_{1:T};\boldsymbol{\mu}_{\mathbf{q}},\boldsymbol{\Sigma}_{\mathbf{q}}) = \mathcal{N}(\mathbf{Y}^{\mathsf{s}};\boldsymbol{\mu}^{\mathsf{s}}_{\mathbf{q}},\boldsymbol{\Sigma}^{\mathsf{s}}_{\mathbf{q}})$$

where Z is a normalisation term. The following relationships exist between the standard and static model parameters

$$\boldsymbol{\Sigma}^{\mathsf{s}\text{-}1}_{\mathbf{q}} = \mathbf{A}^{\mathsf{T}}\boldsymbol{\Sigma}^{\text{-}1}_{\mathbf{q}}\mathbf{A}; \qquad \boldsymbol{\Sigma}^{\mathsf{s}\text{-}1}_{\mathbf{q}}\boldsymbol{\mu}^{\mathsf{s}}_{\mathbf{q}} = \mathbf{A}^{\mathsf{T}}\boldsymbol{\Sigma}^{\text{-}1}_{\mathbf{q}}\boldsymbol{\mu}_{\mathbf{q}} \tag{12}$$

This yields the distribution of the cepstra given a particular component/state sequence. "Synthesis" involves drawing samples from the complete distribution

$$p(\mathbf{Y}^{\mathsf{s}}) = \sum_{\boldsymbol{\omega},\mathbf{q}} P(\boldsymbol{\omega})P(\mathbf{q}|\boldsymbol{\omega})\mathcal{N}(\mathbf{Y}^{\mathsf{s}};\boldsymbol{\mu}^{\mathsf{s}}_{\mathbf{q}},\boldsymbol{\Sigma}^{\mathsf{s}}_{\mathbf{q}}) \tag{13}$$

Sampling from this distribution can be done efficiently using the approaches described in [19]. Once the static sequence has been generated the complete observation sequence $\mathbf{Y}$ is simply obtained using the standard delta and delta-delta expressions.

V. Results

Two continuous digit recognition tasks were used to evaluate the forms of multi-class classification and synthesised data SVM training. The first AURORA 2 is a database where noise has been artificially added to clean data. The second task used in-car data recorded by Toshiba Research Europe Ltd. For both tasks the HTK front-end was used to derive 39 dimensional feature vectors consisting of 12 MFCCs appended with the zeroth cepstrum, delta and delta-delta coefficients. The VTS compensation adopted was similar to the procedure in [15]. An initial hypothesis was generated using a noise model estimated from the first and last 20 frames of each utterance. This hypothesis was used to estimate a per-utterance noise model in an ML-fashion. The final recognition output used this ML-estimated noise model for VTS compensation. For all SVM rescoring experiments the SVMs were built using the top 1500 dimensions of $\phi(\tilde{\mathbf{Y}}_i;\boldsymbol{\lambda})$ ranked using the Fisher ratio and ϵ was set to 2 unless otherwise stated. For the tree-structure and classifier-order see [20] for details.

For both of these continuous digit tasks there was sufficient data to train all classifiers. However, the use of these tasks for the synthesis experiments allows an upper-bound on performance to be obtained. The performance of "real-data" systems can be compared to synthesised approaches. VTS compensated models (using training data estimated noise models) were used to generate "noise" corrupted training data for all words, actual "silence" data was used as this is always available.

A. AURORA 2

AURORA 2 is a small vocabulary digit string recognition task [21]. The utterances in this task are one to seven digits long based on the TIDIGITS database with noise artificially added. The clean training data was used to train the acoustic models. This comprises 8440 utterances from 55 male and 55 female speakers. The acoustic models were 16 emitting states

whole word digit models, with 3 mixtures per state and silence and inter-word pause models. All three test sets, A, B and C, were used for evaluating the schemes. For sets A and B, there were a total of 8 noise conditions (4 in each) at 5 different SNRs, 0dB to 20dB. For test set C there were two additional noise conditions at the same range of SNRs. In addition to background additive noise convolutional distortion was added to test set C. Test set A was used as the development set for tuning parameters.

For the SVM rescoring experiments, the SVMs were trained on a subset of the multi-style training data available for the noise conditions and SNRs in test set A. For each of the noise/SNR conditions there are 422 sentences (a subset of all the training data). For the SVMs training only three of the four available noise conditions (N2-N4) and three of the five SNRs 10dB, 15dB and 20dB were used. This allows the generalisation of the SVM to unseen noise conditions to be evaluated on test set A as well as the test sets B and C. The performance of the baseline system using 1-v-1 classification is given in more detail in [3].

TABLE II
WER (%) AVERAGED OVER 0-20DB FOR TEST SET A (12-CLASS TASK)

System	Cost		WER
	# Class.	# Dec.	(%)
VTS	—	—	9.84
1-v-1	66	66	7.52
Cas (6)	6	1.0	8.66
Cas (17)	17	4.0	7.73
DAG	11	3.6	8.72
FT	11	3.6	8.29
ADAG	66	11	7.54

Table II shows the performance of the standard 1-v-1 classification approach from [3] as well as the cascade approach [3]. As expected the cascade schemes did not perform as well as the full 1-v-1 SVM rescoring scheme, however the computational cost, both in terms of training classifiers and decoding, was less. Using 17 pairs, about 24% of the total number of pairs, 92% of the WER improvement using the 1-v-1 system over the VTS baseline was achieved. For the tree-based approaches Filter-Trees (FT) out-performed the DAG training, illustrating the gains from training the higher level classifiers only on data that can be correctly classified. However neither scheme achieved the same performance as ADAG. Overall for small number of classes, as used here, ADAG appears to be the best form of classifier achieving about the same performance as the 1-v-1 approach.

To initially investigate the synthesis approaches for extended acoustic code-breaking, the whole word acoustic models were used with HMM and HTS synthesis. Here, the acoustic models were adapted to each of the training example noise conditions and used to generate a single static MFCC sequence for each example to train the SVM. To make the training as close as possible to the "real" SVM training scheme the same word-sequence as seen in training was used.

[3]The pairs were selected in terms of individual performance gain on Test Set A. There is thus a slight bias introduced, however similar performance was obtained on Test Sets B and C.

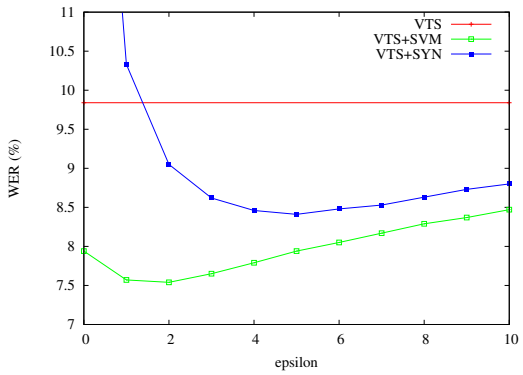

Fig. 2. WER (%) averaged over 0-20dB for Test Set A as ϵ varies for actual (VTS+SVM) and synthesised (VTS+SYN) data

Figure 2 shows the performance of the "real" SVM trained system (VTS+SVM) and the HTS synthesised one (VTS+SYN) on Test Set A as the value of ϵ varies. As expected the value of ϵ that yields the minimum error is higher for the synthesised system than the real system. Another interesting observation is that when $\epsilon = 0$ the performance of the synthesised system is worse than the VTS system. Thus using only the SVM trained on synthesised data is not good, but the approach does yield additional discriminatory information for the VTS-compensated HMM system.

TABLE III
CLEAN, VTS, SVM AND SYNTHESIS (SYN) RESCORING AVERAGED 0-20DB, 1-V-1 MULTI-CLASS CLASSIFICATION.

System	SYN Scheme	ϵ	Test Set WER (%)			Avg (%)
			A	B	C	
VTS	—	—	9.84	9.11	9.53	9.49
+SVM	—	2	7.52	7.35	8.11	7.66
+SYN	HMM	30	9.20	8.51	9.34	9.02
	HTS	5	8.41	8.03	8.70	8.38

The real and synthesised data SVM systems were then run on all the test sets. As an additional contrast data was synthesised using the HMM-based approach to examine whether there was any gain from the more complicated HTS approach. The synthesised system, using $\epsilon = 5$ optimised on Test Set A, gave 60% of the gain from using the real data. In contrast the HMM-based synthesised system, with a large value of $\epsilon = 30$, gave about 25% of the gain. The gains from the HMM synthesised data (where the HMM will be the minimum Bayes' classifier) is possibly to be due to the addition robustness of the maximum margin training in the large score-space.

B. Toshiba In-Car Data

The schemes were also evaluated on a task with real recorded noise: the Toshiba in-car database. This is a corpus collected by Toshiba Research Europe Limited's Cambridge Research Laboratory. It is a small/medium sized task with noisy speech collected in vehicles driving at various conditions. This work used three of the test sets containing digit

sequences (phone numbers). The ENON set, which consists of 835 utterances, was recorded with the engine idle, and has a 35 dB average SNR. The CITY set, which consists of 862 utterances, was recorded driving in cities, and has a 25 dB average SNR. The HWY set, which consists of 887 utterances, was recorded on the highway, and has a 18 dB average SNR. Noise compensation was applied to a speech recogniser trained on clean data from the Wall Street Journal (WSJ) corpus. The total number of states was about 650 with 12 Gaussian components per state. This system is more compact than the usual form of system built on the WSJ data, but is felt to be more realistic for an embedded application whilst maintaining the flexibility to be applicable to a wide-range of tasks. For the initial decoding the acoustic models were decision tree clustered state, cross-word triphones, with three emitting states per HMM, twelve components per GMM and diagonal covariance matrices. Noise corrupted data SVM training and noise models for synthesis were trained using the multi-style training data described in [16].

TABLE IV
VTS, SVM RESCORING USING 1-V-1 AND ADAG MULTI-CLASS CLASSIFICATION AND SYNTHESISED (VTS+SYN, ($\epsilon = 7$)) PERFORMANCE ON THE TOSHIBA IN-CAR TASK.

System	Class. Scheme	Condition WER (%)			Avg (%)
		ENON	CITY	HWY	
VTS	—	1.25	3.09	3.73	2.69
+SVM	1-v-1	1.26	2.60	3.13	2.33
	ADAG	1.27	2.59	3.14	2.33
+SYN	1-v-1	1.22	2.88	3.45	2.52

Table IV shows the performance of the 1-v-1 and ADAG decoding on the Toshiba tasks. The same trends as for the simpler AURORA 2 task were observed. The two multi-class approaches achieve about the same performance showing gains over the standard VTS system. Note these results are all better than multi-style training with/without VTS compensation.

The synthesis results are also shown in table IV. This configuration is closer to the real problem of extending acoustic code-breaking. A context dependent triphone system is being used to generate the data, rather than whole-word models, and the number of components per state is more standard, 12. For this task sequences were initially generated so that there were about 2000 samples per digit, then state/component sequences and finally samples produced. Averaged over the three test set performance gains over the VTS system can be seen. However the value of ϵ required was slightly higher than before, $\epsilon = 7$, with 47% of the gain over the VTS system from the real SVM system obtained.

VI. CONCLUSIONS

This paper has described the investigation into two important issues associated with applying SVM classification, and code-breaking style approaches in general, to speech tasks. The first problem is multi-class classification with SVMs. A range of tree-based approaches and a simple cascade approach were compared to the previously published 1-v-1 classification scheme. Using an adaptive directed acyclic graph approach gave the same performance as 1-v-1 classification, but with reduced run-time computational cost. For larger tasks appropriately selected SVMs run in a cascade approach may be a sensible alternative, as the number of classifiers required does not scale with the number of classes. The second issue addressed was how to build word-based SVM classifiers when the words do not appear in the training data. An HMM-based synthesis approach was found to yield gains over the baseline. However compared to using real data, it is clear that further improvements are still possible. The results given in this paper are preliminary. Though phone-based synthesis has been examined, performance on tasks such as city-names classification still needs to be examined.

ACKNOWLEDGMENT

The authors would like to thank Dr. Heiga Zen for helpful discussions about HTS. Anton Ragni is jointly funded by the EPSRC and Toshiba Research Europe Ltd.

REFERENCES

[1] V. Venkataramani, S. Chakrabartty, and W. Byrne, "Support vector machines for segmental minimum Bayes risk decoding of continuous speech," in *ASRU*, 2003.
[2] V. Vapnik, *Statistical learning theory*. John Wiley & Sons, 1998.
[3] M. Gales and F. Flego, "Discriminative classiers with generative kernels for noise robust speech recognition," in *Proc. ICASSP*, 2009.
[4] N. Smith and M. Gales, "Speech recognition using SVMs," in *Adv. in Neural Inf. Proc. Systems*, 2001.
[5] M. Layton, "Augmented statistical models for classifying sequence data," Ph.D. dissertation, Cambridge University, 2006.
[6] A. Acero, L. Deng, T. Kristjansson, and J. Zhang, "HMM Adaptation using Vector Taylor Series for Noisy Speech Recognition," in *Proc. ICSLP*, Beijing, China, 2000.
[7] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2001.
[8] K. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition," in *Proc. Euro. Symp. Art. Neural Networks*, 1999.
[9] K. Crammer and Y. Singer, "On the algorithmic implementation of multi-class SVMs," *Journal Machine Learning Research*, 2001.
[10] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Adv. in Neural Inf. Proc. Systems 11*, 1999.
[11] A. Beygelzimer, K. Langford, and P. Ravikumar, "Multiclass classification with filter trees," in *ITA Workshop*, February 2007.
[12] B. Kijsirikul and S. Abe, "Multiclass support vector machines using adaptive directed acyclic graphs," in *Int. Joint Conf. Neural Netw.*, 2002.
[13] J. Platt, N. Christianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," in *Adv. in Neural Inf. Proc. Systems 12*, 2000.
[14] K. Tokuda, H. Zen, and A. Black, *Text to speech synthesis: New paradigms and advances*. Prentice Hall, 2004.
[15] J. Li, L. Deng, Y. Gong, and A. Acero, "HMM adaptation with joint compensation of additive and convolutive distortions via vector Taylor series," in *ASRU 2007*, Kyoto, Japan, 2007.
[16] H. Liao and M. Gales, "Joint uncertainty decoding for robust large vocabulary speech recognition," Cambridge University, Tech. Rep. CUED/F-INFENG/TR552, November 2006.
[17] C. Gautier, "Filter trees for combining binary classifiers," M.Phil. dissertation, Cambridge University, 2008.
[18] V. Vural and J. Dy, "A hierarchical method for multi-class support vector machines," in *Proc. ICML*, 2004.
[19] K. Tokuda, H. Zen, and T. Kitamura, "Reformulating the HMM as a trajectory model," in *Proc. Beyond HMM Workshop*, December 2004.
[20] H. AlDamarki, "Filter trees for noise robust small vocabulary speech recognition," M.Phil. dissertation, Cambridge University, 2009.
[21] H.-G. Hirsch and D. Pearce, "The AURORA experimental framework for the evaluation of speech recognition systems under noisy conditions," in *ASR-2000*, 2000.