Support Vector Machines for Speaker and Language Recognition ⋆

W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, P. A. Torres-Carrasquillo

MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420

Abstract

Support vector machines (SVMs) have proven to be a powerful technique for pattern classification. SVMs map inputs into a high dimensional space and then separate classes with a hyperplane. A critical aspect of using SVMs successfully is the design of the inner product, the kernel, induced by the high dimensional mapping. We consider the application of SVMs to speaker and language recognition. A key part of our approach is the use of a kernel that compares sequences of feature vectors and produces a measure of similarity. Our *sequence kernel* is based upon generalized linear discriminants. We show that this strategy has several important properties. First, the kernel uses an explicit expansion into SVM feature space—this property makes it possible to collapse all support vectors into a single model vector and have low computational complexity. Second, the SVM builds upon a simpler mean-squared error classifier to produce a more accurate system. Finally, the system is competitive and complimentary to other approaches, such as Gaussian mixture models (GMMs). We give results for the 2003 NIST speaker and language evaluations of the system and also show fusion with the traditional GMM approach.

Key words: speaker recognition, language recognition, support vector machines

⋆ This work was sponsored by the Department of Defense under Air Force contract F19628-00-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

1 Introduction

A support vector machines (SVM) is a powerful classifier that has gained considerable popularity in recent years. An SVM is a discriminative classifier—it models the boundary between, for example, a speaker and a set of impostors. This approach contrasts to traditional methods for speaker recognition which separately model the probability distributions of the speaker and the general population. By exploring SVM methods, we have several goals—to benchmark the performance of new classification methods for speaker recognition, to gain more understanding of the speaker recognition problem, and to see if SVMs provide complimentary information to traditional GMM approaches. For the final goal, we note that the study of systems which fuse well has been a significant recent effort in the speaker recognition community [1].

Several recent approaches using support vector machines have been proposed in the literature for speech applications. The first set of approaches attempts to model emission probabilities for hidden Markov models [2,3]. This approach has been moderately successful in reducing error rates, but suffers from several problems. First, large training sets result in long training times for support vector methods. Second, the emission probabilities must be approximated [4], since the output of the support vector machine is not a probability. This approximation is needed to combine probabilities using the standard frame independence method used in speaker and language recognition. A second set of approaches tries to combine GMM approaches with SVMs [5,6].

A third set of method is based upon comparing sequences using the Fisher kernel proposed by Jaakkola and Haussler [7]. This approach has been explored for speech recognition in [8]. The application to speaker recognition is

detailed in [9,10]. We propose an alternate kernel [11] based upon generalized linear discriminants [12] and the associated mean-squared error (MSE) training criterion. The advantage of this kernel is that it preserves the structure of generalized linear discriminants [13] which are both computationally and memory efficient. We consider SVMs for two applications in this paper—text-independent speaker and language recognition.

Traditional methods for text-independent speaker recognition are Gaussian mixture models (GMMs) [14], vector quantization [15], and artificial neural networks [15]. Of these methods, GMMs have been the most successful because of many factors, including a probabilistic framework, training methods scalable to large data sets, and high-accuracy recognition.

We also consider language recognition in this paper. Language recognition is a similar problem to speaker recognition in that we are trying to extract information about an entire utterance rather than specific word content. The application of our SVM technique to language recognition shows that our methods are general and have potential applications to several areas in speech.

Many successful approaches to language recognition have been proposed. A classic approach implemented in the parallel-phone recognition language modelling (PPRLM) system of Zissman [16] used phone tokenization of speech combined with a phonotactic analysis of the output to classify the language. A more recent development is the use of methodologies similar to those in speaker recognition. In these approaches, a set of features useful for language recognition have been combined with the GMM to produce excellent recognition performance [17,18]. Our approach to language recognition is based upon features used in the GMM approach.

The outline of the paper is as follows. In Section 2, we introduce the concept of SVMs. Section 3 discusses the overall setup for discriminative training of SVMs. In Section 4, we derive our sequence kernel. We cover the basics of generalized discriminants and then show how they can be incorporated into a sequence kernel. In Section 5, we give a concise algorithmic summary of using our sequence kernel in a speaker or language recognition system. Sections 6 and 7 detail experiments with the resulting system on corpora for the NIST 2003 speaker and language recognition evaluations. In these sections, we also present an approach for fusing our SVM system with a GMM system. Finally, we conclude in Section 8.

2 Support Vector Machines

An SVM [19] is a two-class classifier constructed from sums of a kernel function $K(\cdot, \cdot)$,

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + d, \tag{1}$$

where the t_i are the ideal outputs, $\sum_{i=1}^{N} \alpha_i t_i = 0$, and $\alpha_i > 0$. The vectors $\mathbf{x}_i$ are support vectors and obtained from the training set by an optimization process [20]. The ideal outputs are either 1 or -1, depending upon whether the corresponding support vector is in class 0 or class 1, respectively. For classification, a class decision is based upon whether the value, $f(\mathbf{x})$, is above or below a threshold.

The kernel $K(\cdot, \cdot)$ is constrained to have certain properties (the Mercer condition), so that $K(\cdot, \cdot)$ can be expressed as

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{b}(\mathbf{x})^t \mathbf{b}(\mathbf{y}), \tag{2}$$

where $\mathbf{b}(\mathbf{x})$ is a mapping from the input space (where $\mathbf{x}$ lives) to a possibly

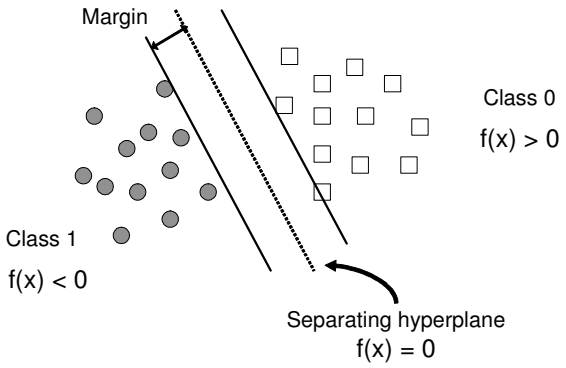

Fig. 1. Support vector machine concept

infinite dimensional space. The kernel is required to be positive semi-definite. The Mercer condition ensures that the margin concept is valid, and the optimization of the SVM is bounded.

The optimization condition relies upon a maximum margin concept, see Figure 1. For a separable data set, the system places a hyperplane in a high dimensional space so that the hyperplane has maximum margin. The data points from the training set lying on the boundaries (as indicated by solid lines in the figure) are the support vectors in equation (1). The focus, then, of the SVM training process is to model the boundary, as opposed to a traditional GMM UBM which would model the probability distributions of the two classes.

3 Discriminative Training for Speaker and Language Recognition

Discriminative training of an SVM for speaker or language recognition is straightforward. Several basic issues must be addressed—handling multiclass data, *world* modelling, and sequence comparison. We handle the first two topics in this section.

We use the following scenarios for speaker and language recognition. For

speaker recognition, we consider two problems—speaker identification and speaker verification. For (closed set) speaker identification, given an utterance, the task is to find the speaker from a list of known individuals. For speaker verification, one is given an utterance and a target model, and the goal is to determine if there is or is not a match. For language recognition, the goal is to determine the language of an utterance from a set of known languages.

Since the SVM is a two-class classifier, we handle speaker recognition and language recognition as verification problems. That is, we use a *one vs. all* strategy. For both closed-set speaker identification and language recognition, we train a target model for the speaker or language respectively. The set of known non-targets are used as the remaining class. Figure 2 shows an example of training an English language model. In the figure, we use English for class 1 data, and the remaining languages are used for class 0 data. This training data is processed with a standard SVM optimizer (we have used SVMTorch [20]) using a kernel, which will be discussed in Section 4. The result is an SVM model that represents English. We repeat the process and produce models for other languages. Speaker identification models are constructed in an analogous fashion with individual speakers substituted for languages. Typically, for both

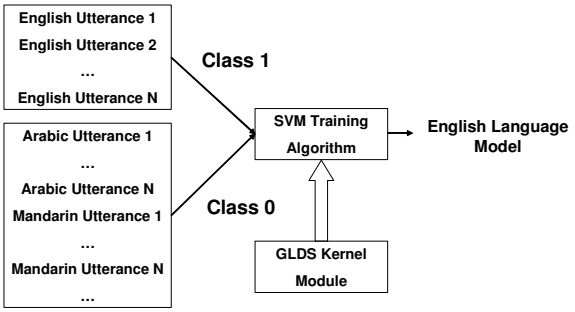

Fig. 2. Training strategy

speaker identification and language recognition, we assume a well-defined set of non-target utterances.

For speaker verification, we train in a manner similar to speaker identification. For each target speaker, we label the target speaker's utterances as class 1. We also construct a *background* speaker set (class 0) that consists of example impostor speakers. The example impostors should be representative of typical impostors to the system. We keep the background speaker set the same as we enroll different target speakers. In contrast to the speaker identification problem, the non-target set of speakers is not as well-defined; we try to capture a representative population of example impostors.

For speaker verification, the support vectors have an interesting interpretation. If $f(x)$ is an SVM for a target speaker, then we can write

$$f(\mathbf{x}) = \sum_{i \in \{i|t_i=1\}} \alpha_i K(\mathbf{x}, \mathbf{x}_i) - \sum_{i \in \{i|t_i=-1\}} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + d. \tag{3}$$

We can think of the first sum as a per-utterance-weighted target score. The second sum has many of the characteristics of a cohort score [21] with some subtle differences. For the second sum, we pick utterances rather than speakers as cohorts. Second, the weighting on these "cohort" utterances is not equal—the cohort score is usually an average of the individual cohorts scores. The interpretation of the SVM score as a cohort normalized score also suggests that we should ensure that our background has a rich speaker set, so that we can always find speakers "close" to the target speaker. Also, note that this interpretation distinguishes the SVM approach from a universal background model method [14], which tries to model the impostor set with one model. Other methods for GMMs including cohort normalization [21] and TNorm [22] are closer to the proposed SVM method; although, the latter method (TNorm)

typically uses a fixed set of cohorts rather than picking our individual speakers.

4 A Sequence Kernel for Speech Applications

4.1 General Structure

To apply an SVM, $f(\mathbf{x})$, to a speaker or language recognition application, we need a method of calculating kernel operations on speech inputs. For recognition, we need a way of taking a sequence of input feature vectors from an utterance, $\{\mathbf{x}_i\}$, and computing the SVM output, $f(\{\mathbf{x}_i\})$. Typically, each vector $\mathbf{x}_i$ would be the cepstral coefficients and deltas for a given frame of speech. One way of handling this situation is to assume that the kernel, $K(\cdot,\cdot)$, in the SVM (1) takes sequences as inputs; i.e., we can calculate $K(\{\mathbf{x}_i\},\{\mathbf{y}_j\})$ for two input sequences $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$. We call this a sequence kernel method.

An alternate method for applying an SVM is to use it as an emission probability estimator in an HMM architecture [2]. Although this second method can yield reasonable results, it has several drawbacks. First, reasonably sized speech problems yield large training sets which can overwelm an SVM training. Second, the SVM output is not a probability, so a framework must be developed for scoring. Finally, working at the frame level gives high overlapping classes yielding a large number of support vectors; this creates large target models and slows scoring. Because sequence kernel methods eliminate these problems, we do not explore this alternate method further.

A challenge in applying the sequence kernel method is deriving a function for comparing sequences. We need a function that, given two utterances, produces a measure of similarity of the speakers or languages. Also, we need a method that is efficient computationally, since we will be performing many kernel

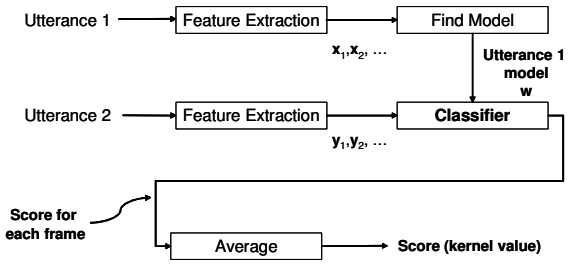

Fig. 3. Sequence kernel

inner products during training and scoring. Finally, the kernel must satisfy the Mercer condition mentioned in Section 2.

Our main idea for constructing a sequence kernel is illustrated in Figure 3. The basic approach is to compare two utterances by training a model on one utterance and then scoring the resulting model on another utterance. This process produces a number that measures the similarity between the two utterances. Two questions that follow from this approach are as follows. 1) Can the train/test process be computed efficiently? 2) Is the resulting comparison a kernel (i.e., does it satisfy the Mercer condition)? We take up these problems in the following sections.

4.2 Generalized Linear Discriminant Scoring

As discussed in Section 3, we can represent our applications as a two class problems; i.e., target and nontarget language or speaker. If ω is a random variable representing the hypothesis, then $\omega = 1$ represents target present and $\omega = 0$ represents target not present.

A score is calculated from a sequence of observations $\mathbf{y}_1, \ldots, \mathbf{y}_n$ extracted from the speech input. The scoring function is based on the output of a generalized linear discriminant function [12] of the form $g(\mathbf{y}) = \mathbf{w}^t \mathbf{b}(\mathbf{y})$, where $\mathbf{w}$ is the vector of classifier parameters (model) and $\mathbf{b}$ is an expansion of the input

space into a vector of scalar functions. An example is

$$\mathbf{b}(\mathbf{y}) = \left[b_1(\mathbf{y}) \; b_2(\mathbf{y}) \; \ldots \; b_{N_e}(\mathbf{y}) \right]^t , \tag{4}$$

where b_i is a mapping from $\mathbb{R}^m$ to $\mathbb{R}$. We typically assume that $b_1(\mathbf{y}) = 1$. Commonly used generalized linear discriminants are polynomials [13] and radial basis functions [23]. Note that we do not use a nonlinear activation function as is common in higher-order neural networks; this allows us to find a closed-form solution for training.

If the classifier is trained with a mean-squared error training criterion and ideal outputs of 1 for $\omega = 1$ and 0 for $\omega = 0$, then $g(\mathbf{y})$ will approximate the *a posteriori* probability $p(\omega = 1|\mathbf{y})$ [23]. We can then find the probability of the entire sequence, $p(\mathbf{y}_1, \ldots, \mathbf{y}_n|\omega = 1)$, as follows. Assuming independence of the observations [24] gives

$$\begin{aligned} p(\mathbf{y}_1, \ldots, \mathbf{y}_n|\omega) &= \prod_{i=1}^{n} p(\mathbf{y}_i|\omega) \\ &= \prod_{i=1}^{n} \frac{p(\omega|\mathbf{y}_i)p(\mathbf{y}_i)}{p(\omega)}. \end{aligned} \tag{5}$$

The scoring method in (5) with scaled posteriors is the same technique as used in the artificial neural network literature for speech applications [25].

For the purposes of classification, we can discard $p(\mathbf{y}_i)$. We take the logarithm of both sides to get the discriminant function

$$d'(\mathbf{y}_1^n|\omega) = \sum_{i=1}^{n} \log \left(\frac{p(\omega|\mathbf{y}_i)}{p(\omega)} \right), \tag{6}$$

where we have used the shorthand $\mathbf{y}_1^n$ to denote the sequence of vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$. We use two terms of the Taylor series of $\log(x) \approx x - 1$ to ob-

tain the final discriminant function

$$d(\mathbf{y}_1^n|\omega) = \frac{1}{n}\sum_{i=1}^{n}\frac{p(\omega|\mathbf{y}_i)}{p(\omega)}. \tag{7}$$

Note that we have discarded the -1 in this discriminant function and normalized by the number of frames since these changes will not affect the classification decision.

There are several reasons for using the Taylor approximation. One reason is that it reduces computation without significantly affecting classifier accuracy. Second, the approximation is not too drastic. A linear approximation is a monotone map, so it preserves score order. Also, we can linearize around any point, a, and get the exact same discriminant function in (7) (scaling and shifting the values of the discriminant function don't change the decision). Typically, the discriminant will have the ratio $p(\omega|\mathbf{y}_i)/p(\omega)$ vary over a fairly small range. Finally, and most importantly, the approximation will symmetrize the role of training and testing utterances and allow us to use the classifier in an SVM framework.

Now assume we have $g(\mathbf{y}) \approx p(\omega = 1|\mathbf{y})$; we call the vector $\mathbf{w}$ the target model. Substituting in the generalized linear discriminant approximation $g(\mathbf{y})$ gives

$$\begin{aligned} d(\mathbf{y}_1^n|\omega = 1) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{w}^t\mathbf{b}(\mathbf{y}_i)}{p(\omega = 1)} \\ &= \frac{1}{np(\omega = 1)}\mathbf{w}^t\left(\sum_{i=1}^{n}\mathbf{b}(\mathbf{y}_i)\right) \\ &= \frac{1}{p(\omega = 1)}\mathbf{w}^t\bar{\mathbf{b}}_y \end{aligned} \tag{8}$$

where we have defined the mapping $\mathbf{y}_1^n \to \bar{\mathbf{b}}_y$ as

$$\mathbf{y}_1^n \to \frac{1}{n}\sum_{i=1}^{n}\mathbf{b}(\mathbf{y}_i). \tag{9}$$

We summarize the scoring method. For a sequence of input vectors $\mathbf{y}_1, \ldots \mathbf{y}_n$ and a target model, $\mathbf{w}$, we construct $\bar{\mathbf{b}}_y$ using (9). We then score using the target model, score $= \mathbf{w}^t \bar{\mathbf{b}}_y$.

4.3 Using Monomials as an Expansion

In this paper, we use monomials as the functions in the expansion (4). A monomial is a polynomial of the form

$$x_{i_1} x_{i_2} \ldots x_{i_k}, \tag{10}$$

where k is less than or equal to the polynomial degree. Here, the input vector $\mathbf{x}$ is

$$\mathbf{x} = \begin{bmatrix} x_1 \; x_2 \; \ldots \; x_m \end{bmatrix}^t. \tag{11}$$

The vector $\mathbf{b}(\mathbf{x})$ is the vector of *all* monomials of the input feature vector (e.g., cepstral coefficients) up to and including degree K. As an example, suppose we have two input features, $\mathbf{x} = \begin{bmatrix} x_1 \; x_2 \end{bmatrix}^t$ and $K = 2$, then the vector is given by

$$\mathbf{b}(\mathbf{x}) = \begin{bmatrix} 1 \; x_1 \; x_2 \; x_1^2 \; x_1 x_2 \; x_2^2 \end{bmatrix}^t. \tag{12}$$

4.4 Generalized Linear Classifier Training

We next review how to train the classifier to approximate the probability $p(\omega|\mathbf{x})$. Let $\mathbf{w}$ be the desired target model. The resulting problem is

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \, \mathbf{E} \left[\left(\mathbf{w}^t \mathbf{b}(\mathbf{x}) - \omega \right)^2 \right], \tag{13}$$

where $\mathbf{E}$ denotes expectation. This criterion can be approximated using the training set as

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{N_{\text{tgt}}} \left| \mathbf{w}^t \mathbf{b}(\mathbf{x}_i) - 1 \right|^2 + \sum_{i=1}^{N_{\text{non}}} \left| \mathbf{w}^t \mathbf{b}(\mathbf{z}_i) \right|^2 \right]. \tag{14}$$

Here, the target training data is $\mathbf{x}_1, \ldots, \mathbf{x}_{N_{\text{tgt}}}$ and the non-target data is $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{non}}}$.

The training method can be written in matrix form. First, define $\mathbf{M}_{\text{tgt}}$ as the matrix whose rows are the expansion of the target's data; i.e.,

$$\mathbf{M}_{\text{tgt}} = \begin{bmatrix} \mathbf{b}(\mathbf{x}_1)^t \\ \mathbf{b}(\mathbf{x}_2)^t \\ \vdots \\ \mathbf{b}(\mathbf{x}_{N_{\text{tgt}}})^t \end{bmatrix}. \tag{15}$$

Define a similar matrix for the nontarget data, $\mathbf{M}_{\text{non}}$. Define

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{\text{tgt}} \\ \mathbf{M}_{\text{non}} \end{bmatrix}. \tag{16}$$

The problem (14) then becomes

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \left\| \mathbf{M}\mathbf{w} - \mathbf{o} \right\|_2, \tag{17}$$

where $\mathbf{o}$ is the vector consisting of N_{tgt} ones followed by N_{non} zeros (i.e., the ideal output).

The problem (17) can be solved using the method of normal equations,

$$\mathbf{M}^t\mathbf{M}\mathbf{w} = \mathbf{M}^t\mathbf{o}. \tag{18}$$

We rearrange (18) to

$$\left(\mathbf{M}^t\mathbf{M}\right)\mathbf{w} = \mathbf{M}_{\text{tgt}}^t\mathbf{1} + \mathbf{M}_{\text{non}}^t\mathbf{0} = \mathbf{M}_{\text{tgt}}^t\mathbf{1}, \tag{19}$$

where $\mathbf{1}$ and $\mathbf{0}$ are the vectors of all ones and all zeros, respectively. If we define $\mathbf{R} = \mathbf{M}^t\mathbf{M}$ and solve for $\mathbf{w}$, then (19) becomes

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{M}_{\text{tgt}}^t\mathbf{1}. \tag{20}$$

4.5 Generalized Linear Discriminant Sequence Kernels

We can now combine the methods from Sections 4.2 and 4.4 to obtain a novel sequence kernel. Combine the target model from (20) with the scoring equation from (8) to obtain the classifier score

$$\text{score} = \frac{1}{p(\omega = 1)}\bar{\mathbf{b}}_y^t\mathbf{w} = \frac{1}{p(\omega = 1)}\bar{\mathbf{b}}_y^t\mathbf{R}^{-1}\mathbf{M}_{\text{tgt}}^t\mathbf{1}. \tag{21}$$

Now $p(\omega = 1) = N_{\text{tgt}}/(N_{\text{non}} + N_{\text{tgt}})$, so that (21) becomes

$$\text{score} = \bar{\mathbf{b}}_y^t\bar{\mathbf{R}}^{-1}\bar{\mathbf{b}}_x, \tag{22}$$

where $\bar{\mathbf{b}}_x$ is $(1/N_{\text{tgt}})\mathbf{M}_{\text{tgt}}^t\mathbf{1}$ (note that this exactly the same as mapping as in (9)), and $\bar{\mathbf{R}}$ is $(1/(N_{\text{non}} + N_{\text{tgt}}))\mathbf{R}$.

The scoring method in (22) is the basis of our sequence kernel. Given two sequences of speech feature vectors, $\mathbf{x}_1^n$ and $\mathbf{y}_1^m$, we compare them by mapping $\mathbf{x}_1^n \to \bar{\mathbf{b}}_x$ and $\mathbf{y}_1^m \to \bar{\mathbf{b}}_y$ and then computing

$$K_{\text{GLDS}}(\mathbf{x}_1^n, \mathbf{y}_1^m) = \bar{\mathbf{b}}_x^t\bar{\mathbf{R}}^{-1}\bar{\mathbf{b}}_y. \tag{23}$$

Note that the function in (23) is not symmetric, so it is not yet a kernel. We discuss several straightforward methods for symmetrizing the kernel in the next section.

After symmetrizing (23), we call K_{GLDS} the **G**eneralized **L**inear **D**iscriminant **S**equence kernel (GLDS is pronounced "golds"). The value $K_{\text{GLDS}}(\mathbf{x}_1^n, \mathbf{y}_1^m)$

can be interpreted as scoring using a generalized linear discriminant on the sequence $\mathbf{y}_1^m$, see (8), with the MSE model trained from feature vectors $\mathbf{x}_1^m$.

4.6 Comments on the GLDS Kernel

Several simplifications and approximations are helpful in using the GLDS kernel in applications. In this section, we point out approximations to $\bar{\mathbf{R}}$, simplifications in training and scoring, and additional general comments on the GLDS kernel.

Two approximations of $\bar{\mathbf{R}}$ are extremely useful in applications with the GLDS kernel. First, consider equation (23). From our derivation, $\bar{\mathbf{R}}$ is dependent on the target data, $\{\mathbf{x}_i\}$. A useful assumption is that, typically, the nontarget data will dominate the calculation of $\bar{\mathbf{R}}$. That is, for $N_{\text{non}} \gg N_{\text{tgt}}$, $\bar{\mathbf{R}} \approx (1/N_{\text{non}})\mathbf{R}_{\text{non}}$. Another way to view this approximation is that we do not need additional target data to approximate the average $\bar{\mathbf{R}}$ if we already have a large nontarget set. A consequence of this approximation is that (23) is now symmetric with respect to the role of the sequences $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$; we can view either as the training or testing sequence. An alternate approach to symmetrization (not used in this paper), is to reverse the role of the two sequences in Figure 3 and then take the average score as the kernel; this operation is equivalent to using an average of the inverse correlation matrices generated in (23).

A second approximation of $\bar{\mathbf{R}}$ that is useful in practice is to calculate only the diagonal of $\bar{\mathbf{R}}$. This dramatically reduces computation since the process is $\mathcal{O}(N_{\text{e}})$ rather than $\mathcal{O}(N_{\text{e}}^2)$, where N_e is the dimension of the expansion (4). We have found in several cases that increasing the dimension of the expansion for polynomials by increasing the degree, see Section 4.3, yielded better accuracy

with less computation than a full correlation $\bar{\mathbf{R}}$.

If $\bar{\mathbf{R}}$ is a full correlation matrix, the computational complexity of training can be dramatically reduced using the following simplification. We factor $\bar{\mathbf{R}}^{-1} = \mathbf{U}^t\mathbf{U}$ using the Cholesky decomposition. Then $K_{\text{GLDS}}(\mathbf{x}_1^n, \mathbf{y}_1^m) = (\mathbf{U}\bar{\mathbf{b}}_x)^t(\mathbf{U}\bar{\mathbf{b}}_y)$. That is, if we transform all the sequence data by $\mathbf{U}\bar{\mathbf{b}}_x$ before training, the sequence kernel is a simple inner product. This method reduces kernel computation from $\mathcal{O}(N_{\text{e}}^2)$ to $\mathcal{O}(N_{\text{exp}})$.

We can simplify scoring with the GLDS kernel with the following technique. Suppose $f(\{\mathbf{x}_i\})$ is the output of the SVM,

$$f(\{\mathbf{x}_i\}) = \sum_{i=1}^{N} \alpha_i t_i \bar{\mathbf{b}}_i^t \bar{\mathbf{R}}^{-1} \bar{\mathbf{b}}_x + d, \tag{24}$$

where the $\bar{\mathbf{b}}_i$ are the support vectors. We can simplify this to

$$f(\{\mathbf{x}_i\}) = \left(\sum_{i=1}^{N} \alpha_i t_i \bar{\mathbf{R}}^{-1} \bar{\mathbf{b}}_i + \mathbf{d} \right)^t \bar{\mathbf{b}}_x, \tag{25}$$

where $\mathbf{d} = \begin{bmatrix} d & 0 & \dots & 0 \end{bmatrix}^t$; we assume that the first entry in the expansion is $b_1(\mathbf{x}) = 1$. In summary, once we train the support vector machine, we can collapse all the support vectors down into a single model $\mathbf{w}$, where

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \bar{\mathbf{R}}^{-1} \bar{\mathbf{b}}_i + \mathbf{d}. \tag{26}$$

Several other items should be mentioned about the GLDS kernel. First, the simplification in (25) gives a very concise way of storing and scoring target models. If we want to search a large database of targets, we can take an input $\{\mathbf{x}_i\}$ and map it to $\bar{\mathbf{b}}_x$ (a single vector). Each target score is then simply an inner product, $\mathbf{w}_{\text{tgt}}^t \bar{\mathbf{b}}_x$ which is $\mathcal{O}(N_{\text{e}})$ operations. Second, another item to note about the GLDS kernel is that it can be incorporated into a text-

dependent speaker or language recognition system. We can create a kernel for each subword or word from an ASR system and then fuse multiple kernels with different weights to create a new scoring function. This approach is discussed in a hybrid SVM/HMM system in [26]. Third, we mention that the GLDS kernel is an explicit expansion into SVM feature space; i.e., we are not using the kernel trick common in the SVM literature [19]. Using an explicit expansion makes it possible to compact the model as given in (25) resulting in considerable reduction in computation for scoring and model storage.

5 Algorithms for the GLDS Kernel

After deriving the mathematics behind the GLDS kernel in Section 4, we now discuss a basic algorithmic framework for using the GLDS kernel. We make several assumptions to simplify the presentation. First, we will assume that we are performing speaker verification. Second, we assume that the matrix $\bar{\mathbf{R}}$ in (23) is approximated using nontarget data and a diagonal structure as discussed in 4.6. These simplifying assumptions make it possible to split the training process into two parts: 1) background creation, and 2) target speaker training.

Table 1 shows the process of background training for the SVM GLDS kernel. As mentioned in Section 3, the background should be a "large" corpus representative of the expected impostors to the system. The result of background creation is a set of vectors, $\{\bar{\mathbf{b}}_z^i\}$, that can be used in the SVM training process as the class will ideal output -1. Several notational items should be mentioned from Table 1. First, the notation $\mathbf{z} = \mathbf{x}.*\mathbf{y}$ means $\mathbf{z}$ is the vector $z_i = x_i * y_i$. Similarly the square root of a vector is the square root of its entries.

Table 1
Creating a nontarget background

1) Given: N_{utt} nontarget utterances
2) $N_{\text{tot}} = 0$
3) $\mathbf{r} = \mathbf{0}$
4) For $i = 1$ to N_{utt}
   5) Let $\{\mathbf{z}_i\}$, $i = 1, \ldots, N_z$, be the features extracted from the ith nontarget utterance
   6) Calculate and store $\bar{\mathbf{b}}_z^i = (1/N_z) \sum_{i=1}^{N_z} \mathbf{b}(\mathbf{z}_i)$
   7) $\mathbf{r} = \mathbf{r} + \sum_{i=1}^{N_z} \mathbf{b}(\mathbf{z}_i).*\mathbf{b}(\mathbf{z}_i)$
   8) $N_{\text{tot}} = N_{\text{tot}} + N_z$
9) Next i
10) Let $\mathbf{r} = (1/N_{\text{tot}})\mathbf{r}$
11) Let $\mathbf{r}_{\text{sqrt}} = 1./\sqrt{\mathbf{r}}$
12) For all $i = 1, \ldots, N_{\text{utt}}$, replace $\bar{\mathbf{b}}_z^i = \mathbf{r}_{\text{sqrt}}.*\bar{\mathbf{b}}_z^i$.
13) The set of vectors $\{\bar{\mathbf{b}}_z^i\}$ is the nontarget background

Table 2
Creating a target model

1) Given: N_{tgt} target utterances
2) For $i = 1$ to N_{tgt}
   3) Let $\{\mathbf{x}_i\}$, $i = 1, \ldots, N_x$, be the features extracted from the ith target utterance
   4) $\bar{\mathbf{b}}_x^i = (1/N_x) \sum_{i=1}^{N_x} \mathbf{b}(\mathbf{x}_i)$
   5) $\bar{\mathbf{b}}_x^i = \mathbf{r}_{\text{sqrt}}.*\bar{\mathbf{b}}_x^i$ where $\mathbf{r}_{\text{sqrt}}$ is from the background training algorithm in Table 1
6) Next i
7) Train an SVM using: a linear kernel ($K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t\mathbf{y}$), ideal outputs of 1 for $\{\bar{\mathbf{b}}_x^i\}$, and ideal outputs of -1 for $\{\bar{\mathbf{b}}_z^i\}$ (computed in Table 1). For the trained SVM, call the resulting weights, α_i, the support vectors, $\mathbf{b}_i$, and the constant, d.
8) Compute the target model as $\mathbf{w} = \mathbf{r}_{\text{sqrt}}.* \left(\sum_{i=1}^{l} \alpha_i t_i \bar{\mathbf{b}}_i \right) + \mathbf{d}$ where $\mathbf{d} = \begin{bmatrix} d & 0 & \ldots & 0 \end{bmatrix}^t$, and t_i is the ideal output for the ith support vector.

After creating a background for the speaker verification, we can now train target models. The basic process is shown in Table 2. The result of training is a target model, $\mathbf{w}$. Note that the algorithm in Table 2 requires no special SVM training tool—one can use any SVM tool that implements a linear kernel for classification. Typically, we have used SVMTorch [20].

After we obtain target models from the training process in Table 2, we can then score with these models in a straightforward manner. Given an input utterance, we convert it to a sequence of feature vectors, $\{\mathbf{y}_j\}$, and then to an

average expansion, $\bar{\mathbf{b}}_y$. The output score is $s = \mathbf{w}^t \bar{\mathbf{b}}_y$. Since we have included the matrix $\bar{\mathbf{R}}^{-1}$ in the model, we don't need to apply it to $\bar{\mathbf{b}}_y$.

6 Speaker Recognition Experiments

6.1 The NIST 2003 Speaker Recognition Evaluation

The NIST 2003 speaker recognition evaluation (SRE) included multiple tasks for both one- and two- speaker detection. For the purposes of this paper, we focus on the one speaker detection task from limited data.

The data in the one-speaker limited-data detection task was taken from the second release of the cellular Switchboard corpus of the Linguistic Data Consortium. Training data was nominally 2 minutes of speech from a target speaker excerpted from a single conversation. The training corpus contained 356 target speakers. Each test segment contained a single speaker. The primary task was detection of the speaker from a segment of length 15 to 45 seconds. The test set had 2,215 true trials and 25,945 false trials (impostor attempts). For evaluation, NIST used the decision cost function

$$\begin{aligned} C_{\text{det}} = & C_{\text{miss}} P(\text{miss}|\text{target}) P(\text{target}) + \\ & C_{\text{FA}} P(\text{FA}|\text{nontarget}) P(\text{nontarget}) \end{aligned} \tag{27}$$

as well as reporting standard measures such as equal error rate (EER). In (27), $C_{\text{miss}} = 10$, $C_{\text{FA}} = 1$ and $P(\text{target}) = 0.01$. More details on the evaluation may be found in [27].

6.2 SVM setup

We used two different sets of features for the SVM to explore performance. Linear prediction cepstral coefficients (LPCCs) were extracted using a configuration from [13]. The mel-frequency cepstral coefficient (MFCC) configuration

was based on the best feature set for a GMM implementation used in the NIST speaker recognition evaluations.

LPCC front end processing. LPCC feature extraction is performed using a 30 ms window with a rate of 100 frames/second. A Hamming window is applied, and then 12 LP coefficients are extracted. From 12 LP coefficients, 18 cepstral coefficients (LPCCs) are calculated. Deltas are extracted from the 18 LPCCs. This results in a feature vector of dimension 36 (18 LPCCs and deltas). Energy-based speech activity detection is used to remove nominally nonspeech frames. Both mean and variance normalization are applied to produce zero mean, unit variance features.

MFCC front end processing. A 19-dimensional MFCC vector is extracted from the pre-emphasized speech signal every 10 ms using a 20 ms Hamming window. The mel-cepstral vector is computed using a simulated triangular filterbank on the DFT spectrum. Bandlimiting is performed by retaining only the filterbank outputs from the frequency range 300 Hz–3140 Hz. Cepstral vectors are processed with RASTA filtering to mitigate linear channel bias effects. Delta-cepstral coefficients are then computed over a ± 2 frame span and appended to the cepstra vector, producing a 38 dimensional feature vector. The feature vector stream is processed through an adaptive, energy-based speech detector to discard low-energy vectors. Finally, both mean and variance normalization are applied to the individual features.

Training. The SVM uses a GLDS kernel with an expansion into feature space with a monomial basis. All monomials up to degree 3 are used, resulting in a feature space expansion of dimension 9139 for the LPCC features and dimension 10,660 for the MFCC features. We use a diagonal approximation

to the kernel inner product matrix. A *background* for the SVM consists of a set of speakers taken from a corpus not used in the train/test set. The NIST SRE 01 evaluation is used as a background. SVM training is performed as a two-class problem, where all of the speakers in the background have SVM target -1 and the current speaker under training has SVM target +1. For each conversation in the background and for the current speaker under training, an average feature expansion is created. SVM training is then performed using the GLDS kernel implemented using SVMTorch.

Scoring. For each utterance, the standard front end is used. An average feature expansion is then calculated. Scores for each target speaker are an inner product between the speaker model and the average expansion. A gender T-norm score is also computed using 100 males and 100 females from the NIST SRE 2001 task; details on T-norm may be found in [28].

6.3 Experiments

Figure 4 shows the DET plot of the SVM system applied to the one-speaker NIST SRE 2003 limited data task. The two systems differ only in the front end processing—SVM-M uses MFCC features, and SVM-L uses LPCC features. Both systems are performing well compared with standard approaches—see the next section.

6.4 Fusing the SVM GLDS system with a GMM system

We fused the SVM GLDS kernel with a standard GMM system for speaker recognition. The goals were twofold. First, we wanted to show how the new SVM approach compared to the standard GMM approach. Second, we wanted to explore fusion of GMMs and SVMs.

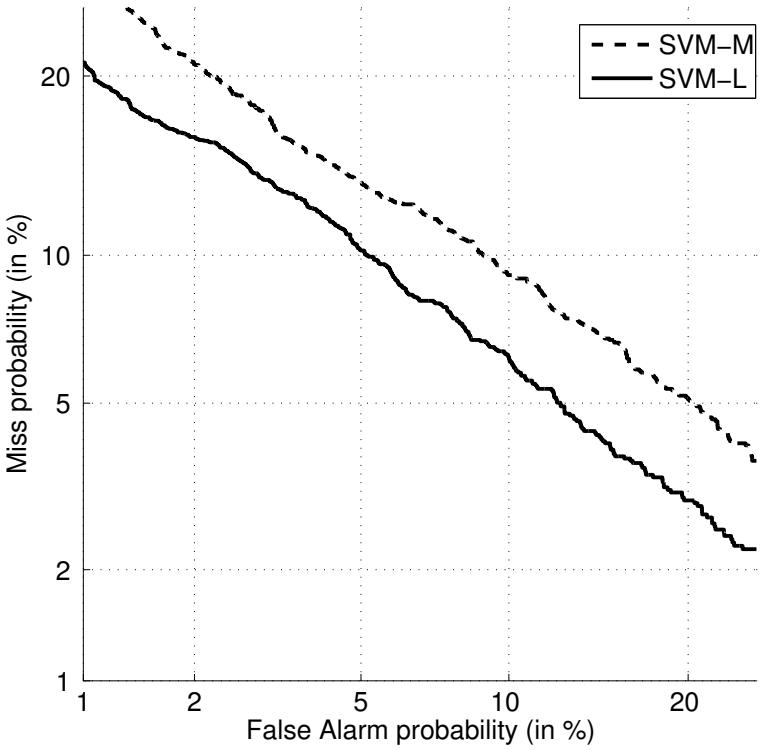

Fig. 4. SVM speaker recognition on the NIST SRE 2003 1sp limited data task

GMM feature extraction. The GMM feature extraction process was the same as the MFCC feature extraction given in Section 6.2 except for one additional step—feature mapping. After producing MFCC features, feature mapping is applied to help remove channel effects [29]. Briefly, the feature mapper works as follows. A channel-independent root model is trained using all available channel-specific data. Next, channel-specific models are derived by using MAP adaptation of root parameters with channel-specific data. For an input utterance, the most likely channel specific model is first identified then each feature vector in the utterance is shifted and scaled using the top-1 scoring mixture parameters in the root and channel-specific models to map the feature vector to the channel-independent feature space. Ten channel models derived from Switchboard landline and cellular corpora were used.

GMM training and scoring. The basic system used is a likelihood ratio detector with target and alternative probability distributions modeled by GMMs. Target models are derived by Bayesian adaptation (a.k.a. MAP estimation) of the UBM parameters using the designated training data [14]. Based on observed better performance, only the mean vectors are adapted. The amount of adaptation of each mixture mean is data dependent with a relevance factor of 16 used. Gender dependent T-norming [22] was applied to the final scores; speakers are taken from the Switchboard 2 part 1 corpus (100 per gender).

6.5 Speaker Recognition Fusion Results

We performed experiments on the 2003 NIST SRE evaluation data described in Section 6.1. Fusion of different systems is accomplished using *equal* linear weighting of the different systems scores; i.e., if two systems produce scores, s_1 and s_2, then the fused score is $s = 0.5s_1 + 0.5s_2$. Since all systems use T-norm, no further normalization of scores is required.

Figure 5 and Table 3 show the results of fusion. In the table, minDCF stands for *minimum decision cost function* where the cost function is given by (27). In the figure, SVM-L is the SVM with LPCC features, and SVM-M is the SVM with MFCC features. Both the figure and the table show that the SVM and GMM fuse in a complementary way reducing error rates substantially. An interesting and important fact shown in the figure is that gains in performance are due both to different features (LPCC and MFCC) and the different speaker modelling techniques (SVM and GMM). For the NIST 2003 corpus, we have found that the SVM performs best with LPCC features. It is not clear whether this property is due to interactions with the SVM modelling (e.g., our diagonal correlation approximation) or a corpus idiosyncrasy. Certainly, our MFCC

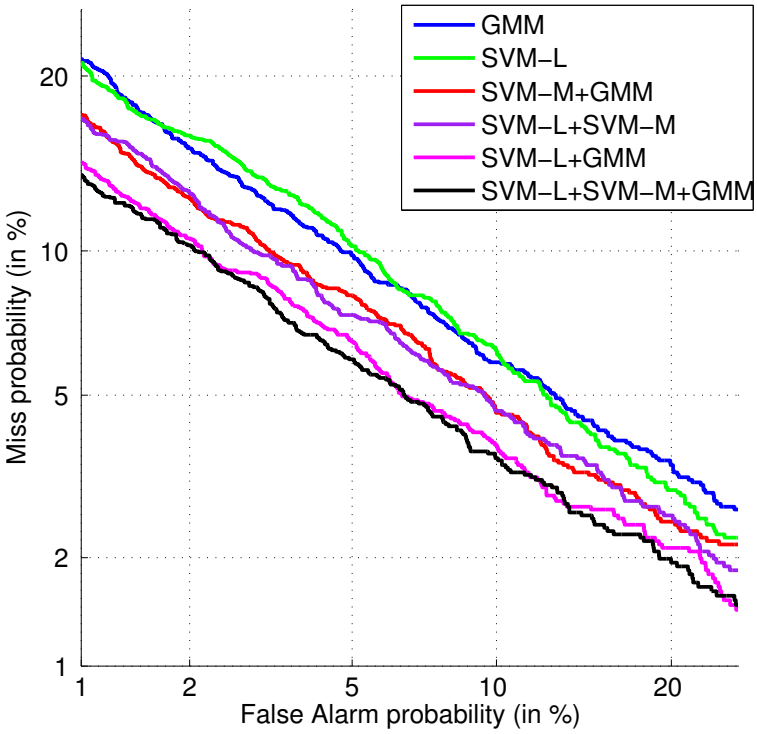

Fig. 5. NIST 2003 1sp limited data fusion results

feature extraction has been tuned for a GMM; further research into optimizing features for the SVM approach should be explored.

Another point to make about Figure 5 and Table 3 is the relative performance of the GMM and SVM. The GMM system uses a background data set, features (MFCCs), and TNorm which have been extensively optimized for performance. The SVM feature sets and methods presented are some initial explorations into the best configuration. If we compare the best SVM system, SVM-L, with the GMM system, the error rates are close—7.72% and 7.47%, respectively. This result shows that the SVM is competitive with the GMM for this set of experiments. Further research is needed to fully understand the performance of the new SVM system relative to the GMM system.

Table 3
Comparison of EER and minDCF for different systems on the 2003 NIST SRE 1sp limited data evaluation

System	EER	minDCF
GMM	7.47 %	0.0306
SVM-L	7.72 %	0.0303
SVM-M	9.57 %	0.0374
SVM-M+GMM	6.74 %	0.0266
SVM-L+SVM-M	6.46 %	0.0262
SVM-L+GMM	5.73 %	0.0237
SVM-L+SVM-M+GMM	5.55 %	0.0230

7 Language Recognition Experiments

7.1 Features for Language Recognition

One of the significant advances in performing language recognition using GMMs was the discovery of a better feature set for language identification [17]. The improved feature set, shifted delta cepstral (SDC) coefficients, are an extension of delta-cepstral coefficients. Prior to the use of SDC coefficients, GMM-based language recognition was less accurate than alternate approaches [16]. SDC coefficients capture variation over many frames of data; e.g., our current approach uses 20 consecutive frames of cepstral coefficients. This long term analysis might explain the effectiveness of the SDC features in capturing language specific information.

SDC coefficients are calculated as shown in Figure 6. SDC coefficients are based upon four parameters, typically written as N-d-P-k. For each frame of data, MFCCs are calculated based on N; i.e., c_0, c_1, ..., c_{N-1} (note that c_0 is used). The parameter d determines the spread over which deltas are calculated, and the parameter P determines the gaps between successive delta computations. For a given time, t, we obtain

$$\Delta c(t,i) = c(t+iP+d) - c(t+iP-d) \tag{28}$$

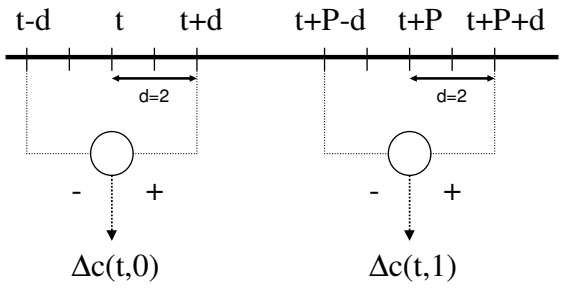

Fig. 6. Shifted delta cepstral coefficients

as an intermediate calculation. The SDC coefficients are then k stacked versions of (28),

$$SDC(t) = \left[\Delta c(t,0)^t \ \Delta c(t,1)^t \ \ldots \ \Delta c(t,k-1)^t\right]^t . \tag{29}$$

7.2 2003 NIST Language Recognition Evaluation

In 2003, NIST held an evaluation to assess the current performance of language recognition systems for conversational telephone speech. The basic task of the evaluation was to detect the presence of a hypothesized target language given a segment of speech. The target languages were American English, Egyptian Arabic, Farsi, Canadian French, Mandarin, German, Hindi, Japanese, Spanish, Korean, Tamil, and Vietnamese. Evaluation of the task was performed through standard measures: a decision cost function and EER.

The training, development, and test data were primarily drawn from the Call-Friend corpus available from the Linguistic Data Consortium (LDC). Training data consisted of 20 complete conversations (nominally 30 minutes) for each of the 12 target languages. Development data was drawn from the 1996 NIST LID development and evaluation sets. Test data consisted of speech segments of length 3, 10, and 30 seconds. For each of these durations, 960 true trials and 10,560 false trials were produced from the primary evaluation task. Per-

formance was measure by EER and the detection cost function given in (27) with $C_{\text{miss}} = C_{\text{FA}} = 1$ and $P_{\text{target}} = 0.5$. For more information, we refer to the NIST evaluation plan [30,31].

7.3 Experiments

Experiments are performed using the NIST LRE evaluation data and the primary evaluation condition. We focus on language detection for the 30 second case. This resulted in 960 true trials and 10,560 false trials.

For the SVM system, SDC features are extracted as in Section 7.1. Our primary representation *N-d-P-k* is 7-1-3-7. This representation is selected based upon prior excellent results with this choice [17,32]. After extracting the SDC features, nonspeech frames are eliminated, and each feature is normalized to mean 0 and variance 1 on a per-utterance basis. This results in a sequence of features vectors of dimension 49 for each utterance.

The SVM system uses the GLDS kernel, as described in Section 4, with a diagonal correlation matrix $\bar{\mathbf{R}}$. All monomials up to degree 3 are used in the expansion $\mathbf{b}(\mathbf{x})$; this results in an expansion dimension of 22,100.

The performance of language recognition is enhanced considerably by applying backend processing to the target language scores. A simple backend process is to apply a log-likelihood normalization. Suppose $s_1, \ldots, s_M$ are the scores from the M language models for a particular message. To normalize the scores, we find new scores, s'_i given by

$$s'_i = s_i - \log\left(\frac{1}{M-1}\sum_{j\neq i} e^{-s_j}\right) \tag{30}$$

A more complex full backend process is given in [16,32]; this process trans-

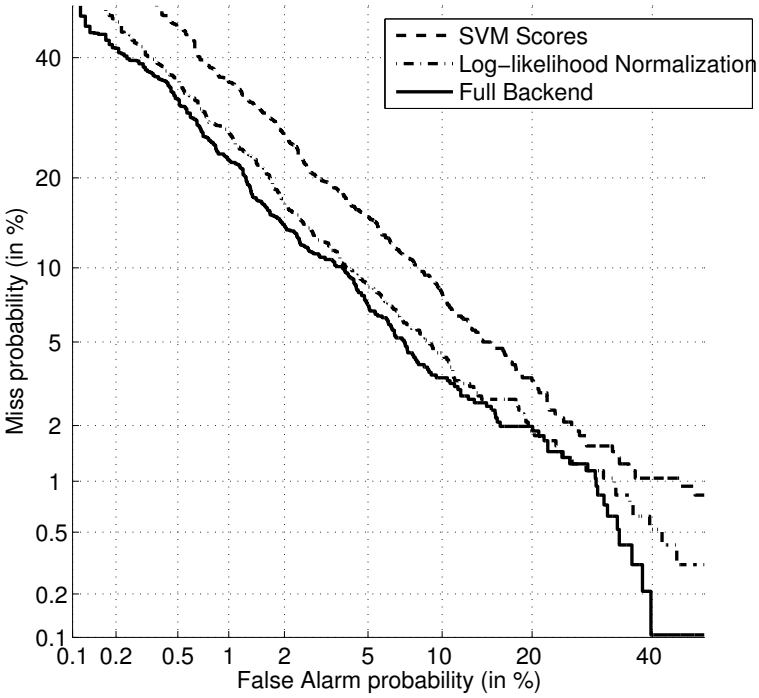

Fig. 7. SVM language recognition on the NIST LRE 2003 30s task

forms language scores with LDA, models the transformed scores with diagonal covariance Gaussians (one per language), and then applies the transform in (30).

Figure 7 shows the performance of the SVM on the NIST LRE 2003 30 second task. In the figure, we compare the performance of three systems. As can be seen, the "raw" SVM scores (i.e., no backend normalization) perform considerably worse than a backend processed score. If we do only LLR normalization as in (30) on the SVM scores, this performs substantially better. Finally, using the full backend process described performs the best.

7.4 Fusing with a GMM-based Language Recognition System

We compare and fuse our SVM system with a GMM language recognition system. The GMM system setup and description are given in [32]. Briefly, each language model consisted of a GMM with 2048 mixture components. SDC

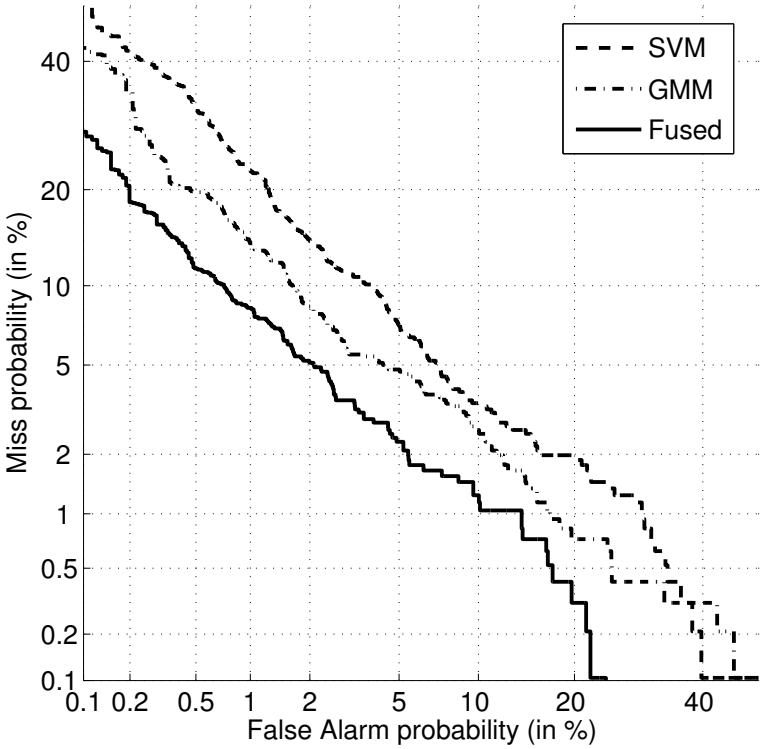

Fig. 8. Performance of three different systems on the NIST 2003 language recognition evaluation for 30s duration tests

features were extracted using the parameter specification 7-1-3-7; the features were postprocessed using the feature mapping technique [29]. Language models were gender dependent, so a total of 24 models were used for the 12 target languages.

We considered the performance of the system relative to a GMM language recognition system, see Figure 8. In the figure, we see that the new SVM system is performing competitively with the state-of-the-art GMM system. The figure also shows the fusion of the two systems. Fusion was accomplished with a backend fuser described in [16,32]. As the figure illustrates, the fusion combination works extremely well, significantly outperforming both individual systems. The EERs for these different systems is shown in Table 4.

Table 4
EER performance of the systems for the 30s test

System	EER
SVM	6.1%
GMM	4.8%
Fused	3.2%

8 Conclusions

We have introduced a new technique for speaker and language recognition based upon SVMs. A novel sequence kernel was derived called the generalized linear discriminant sequence (GLDS) kernel. This kernel was shown to be computationally efficient and easily incorporated into standard SVM packages. We applied this new SVM approach to the NIST 2003 speaker and language evaluation. The results demonstrated the accuracy and success of the approach. Finally, the SVM was compared and fused with a GMM system. The SVM was shown to perform comparably to the GMM in EER and minDCF performance. Additionally, the SVM was shown to provide complementary scoring information resulting in substantially lower error rates when it was fused with a GMM system.

References

[1] J. P. Campbell, D. A. Reynolds, R. B. Dunn, Fusing high- and low-level features for speaker recognition, in: Proc. Eurospeech, 2003, pp. 2665–2668.

[2] V. Wan, W. M. Campbell, Support vector machines for verification and identification, in: Neural Networks for Signal Processing X, Proceedings of the 2000 IEEE Signal Processing Workshop, 2000, pp. 775–784.

[3] A. Ganapathiraju, J. Picone, Hybrid SVM/HMM architectures for speech recognition, in: Speech Transcription Workshop, 2000.

[4] J. C. Platt, Probabilities for SV machines, in: A. J. Smola, P. L. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, The MIT Press, 2000, pp. 61–74.

[5] J.Kharroubi, D. Petrovska-Delacretaz, G. Chollet, Combining GMMs with support vector machines for text-independent speaker verification, in: Eurospeech, 2001, pp. 1757–1760.

[6] J. Kharroubi, D. Petrovska-Delacretaz, G. Chollet, Text-independent speaker verification using support vector machines, in: Proc. Speaker Odyssey, 2001, pp. 51–54.

[7] T. S. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, in: M. S. Kearns, S. A. Solla, D. A. Cohn (Eds.), Advances in Neural Information Processing 11, The MIT Press, 1998, pp. 487–493.

[8] N. Smith, M. Gales, M. Niranjan, Data-dependent kernels in SVM classification of speech patterns, Tech. Rep. CUED/F-INFENG/TR.387, Cambridge University Engineering Department (2001).

[9] S. Fine, J. Navrátil, R. A. Gopinath, A hybrid GMM/SVM approach to speaker

recognition, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 2001.

[10] V. Wan, S. Renals, SVMSVM: support vector machine speaker verification methodology, in: Proceedings of the International Conference on Acoustics Speech and Signal Processing, 2003, pp. 221–224.

[11] W. M. Campbell, Generalized linear discriminant sequence kernels for speaker recognition, in: Proceedings of the International Conference on Acoustics Speech and Signal Processing, 2002, pp. 161–164.

[12] C. M. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.

[13] W. M. Campbell, K. T. Assaleh, Polynomial classifier techniques for speaker verification, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 1999, pp. 321–324.

[14] D. A. Reynolds, T. F. Quatieri, R. Dunn, Speaker verification using adapted Gaussian mixture models, Digital Signal Processing 10 (1-3) (2000) 19–41.

[15] K. R. Farrell, R. J. Mammone, K. T. Assaleh, Speaker recognition using neural networks and conventional classifiers, IEEE Trans. on Speech and Audio Processing 2 (1) (1994) 194–205.

[16] M. Zissman, Comparison of four approaches to automatic language identification of telephone speech, IEEE Trans. Speech and Audio Processing 4 (1) (1996) 31–44.

[17] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, J. R. Deller, Jr., Approaches to language identification using Gaussian mixture models and shifted delta cepstral features, in: International Conference on Spoken Language Processing, 2002, pp. 89–92.

[18] E. Wong, J. Pelecanos, S. Myers, S. Sridharan, Language identification using efficient Gaussian mixture model analysis, in: Australian International Conference on Speech Science and Technology, 2000.

[19] N. Cristianini, J. Shawe-Taylor, Support Vector Machines, Cambridge University Press, Cambridge, 2000.

[20] R. Collobert, S. Bengio, SVMTorch: Support vector machines for large-scale regression problems, Journal of Machine Learning Research 1 (2001) 143–160.

[21] A. E. Rosenberg, J. DeLong, C.-H. Lee, B.-H. Juang, F. K. Soong, The use of cohort normalized scores for speaker verification, in: Proceedings of the International Conference on Spoken Language Processing, 1992, pp. 599–602.

[22] R. Auckenthaler, M. Carey, H. Lloyd-Thomas, Score normalization for text-independent speaker verification systems, Digital Signal Processing 10 (2000) 42–54.

[23] J. Schürmann, Pattern Classification, John Wiley and Sons, Inc., 1996.

[24] L. Rabiner, B.-H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, 1993.

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

[26] W. M. Campbell, A SVM/HMM system for speaker recognition, in: Proceedings of the International Conference on Acoustics Speech and Signal Processing, 2003, pp. II–209–212.

[27] M. Przybocki, A. Martin, The NIST year 2003 speaker recognition evaluation plan, http://www.nist.gov/speech/tests/spk/2003/index.htm (2003).

[28] W. M. Campbell, D. A. Reynolds, J. P. Campbell, Fusing discriminative and generative methods for speaker recogntion: Experiments on Switchboard and

NFI/TNO field data, in: Proc. Odyssey Speaker and Language Workshop, 2004, pp. 41–44.

[29] D. A. Reynolds, Channel robust speaker verification via feature mapping, in: Proceedings of the International Conference on Acoustics Speech and Signal Processing, Vol. 2, 2003, pp. II–53–56.

[30] The 2003 NIST language recognition evaluation plan, http://www.nist.gov/speech/tests/lang/index.htm (2003).

[31] A. F. Martin, M. A. Przybocki, NIST 2003 language recognition evaluation, in: Proceedings of Eurospeech, 2003, pp. 1341–1344.

[32] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell, D. A. Reynolds, Acoustic, phonetic, and discriminative approaches to automatic language identification, in: Proceedings of Eurospeech, 2003, pp. 1345–1348.