PARAMETERIZATION OF PROSODIC FEATURE DISTRIBUTIONS FOR SVM MODELING IN SPEAKER RECOGNITION

Luciana Ferrer[1] *Elizabeth Shriberg*[2] *Sachin Kajarekar*[2] *Kemal Sönmez*[2]

[1]Department of Electrical Engineering, Stanford University, Stanford, CA, USA
[2]Speech Technology and Research Laboratory, SRI International, Menlo Park, CA, USA

ABSTRACT

Multiple recent studies have shown that speaker recognition performance using frame-based cepstral features is improved by adding higher-level information, including prosodic and lexical features. This paper explores the important question of finding a good kernel for a system that models syllable-based prosodic features using support vector machines (SVMs). The system has been the best performing of our high-level systems in the last two NIST evaluations, and gives significant improvements when combined with cepstral-based systems. We introduce two new methods for transforming the syllable-level features into a single high-dimensional vector that can be well modeled by SVMs, resulting in significant gains in speaker recognition performance.

Index Terms— Speaker recognition, Prosody, GMM, SVM

1. INTRODUCTION

We consider the task of text-independent speaker verification, i.e., given a sample from a speaker and a claimed identity we need to decide whether the claim is true or false. In the last few years a common approach to speaker verification has been to combine different knowledge sources by modeling them separately, combining them at the score level to produce the final score that is later thresholded to obtain a decision. In this paper we focus on a high-level system, which we call SNERFs, a special case of Non-Uniform Extraction Region Features (NERFs) with extraction regions defined by the syllables in the utterance [1]. We have found this system to be the best performing of all our high-level systems on the last two speaker recognition evaluations (SRE) organized by NIST every year. Furthermore, the system gives significant improvements when combined with state-of-the-art cepstral-based systems like the GMM-UBM and MLLR-SVM [2].

Recently, SVMs have been found to be a very effective method for modeling feature vectors for speaker recognition [3] [4], in particular in cases where the feature vectors are high-dimensional and/or sparse. When using SVMs for modeling, a major concern is to find a transformation of the feature vectors that results in good discrimination between the two classes (impostors versus true speakers). This transformation is usually realized through the use of kernel functions, as in [4] and [5]. In this work we present and compare different methods (which can be thought of as highly non linear kernel functions) for turning SNERFs into a representation that can be successfully modeled using SVMs.

Since much of the information about the speaker identity is conveyed on the way the features develop over time, our proposed representation focuses on the modeling of time sequences of these features along with their individual distributions. Furthermore, prosodic features present particular challenges in that their distributions can be continuous, discrete or mixed and that they can be undefined. The methods introduced in this paper are capable of modeling features with these characteristics and can be applied to any binary classification task in which features of this nature are used.

The remainder of this paper is organized as follows: Section 2 briefly describes the features, Section 3 introduces the transformations proposed in this paper, Section 4 presents the experiments and results, and Section 5 gives the conclusions.

2. PROSODIC FEATURES

The features used in this work are described in detail in [1]. Briefly, a feature vector is obtained for each syllable in the utterance. Syllables are obtained automatically from the recognition alignments. For each syllable we extract a large number of features related to the duration, pitch, and energy of the syllable. Currently, the dimension of the feature vector is 140. An important characteristic of these features is that they may be undefined. For example, all pitch features are undefined when no pitch was found during a syllable. This makes the modeling of these features more challenging since the fact that a feature is undefined may contain information about the speaker and we may want to include that information in the model. Another characteristic that makes these features particularly hard to work with is that they can be either discrete or continuous, or even a mix of both. Examples of discrete features are the raw (un-normalized) duration features which measure time in frames, as given by the recognition alignments. These features can take only integer values from 0 to a few tens. Examples of continuous features are the maximum value of the median-filtered pitch over the syllable, or the average normalized energy over the syllable's nucleus. Normalized counts are examples of mixed distribution features, where we find a peak at 0 and a continuous distribution on the positive values.

3. FEATURE DISTRIBUTION PARAMETERIZATION

To model a set of features using SVMs we may first need to define a transformation (which in turn defines a kernel) that can convert them into new features that can be well classified by a hyperplane. See, for example, [4] for a brief explanation of SVMs and further references. In speaker recognition, we are given a set of samples from each target speaker that we use as positive examples for the SVM training. The negative examples are given by samples of speech from a separate set of speakers, none of which should be included among our target speakers.

In the following sections we describe three different ways of transforming the set of syllable-level feature vectors $X = \{x^{(1)}, x^{(2)}, ..., x^{(T)}\}$ into a single sample-level vector $b(X)$ that will be input to the SVMs. Since we train linear kernel SVMs, the whole process is equivalent to using a kernel given by $K(X, Y) =$

$b(X)^t b(Y)$. Each component of X corresponds to either a syllable or a pause. We call each of these components a slot t. If the slot corresponds to a syllable, $x^{(t)}$ contains the vector of prosodic features $f^{(t)}$ for that syllable; if it corresponds to a pause, $x^{(t)}$ contains the pause length $p^{(t)}$. The overall idea is to make a representation of the distribution of the features and then use the parameters of that representation to form the vector $b(X)$. In all cases we consider each prosodic feature separately, generating models for the distribution of the features in all syllables and also for the distribution of the features in two and three consecutive syllables (we call these unigrams, bigrams and trigrams, respectively). This allows us to model change in the features over time. Furthermore, we create separate models for sequences including pauses in different positions of the sequence. For each N-gram length, each feature and each pattern of pause/non-pause we obtain a separate model. For example, for trigrams we obtain five different models: (S, S, S), (P, S, S), (S, P, S), (S, S, P), (P, S, P) for each feature. Each pair {feature, pattern} determines what we will call a *token*. The parameters corresponding to all tokens are concatenated to obtain $b(X)$. The following sections describe three different parameterizations of the token distributions.

Prior to SVM training and testing, the components of the vector $b(X)$ need to be normalized to equalize their dynamic ranges. To this end, we apply rank normalization, replacing each feature value by its rank on a held-out set of samples, and then scaling the ranks to a value between 0 and 1. We have found this normalization method to be consistently better than variance normalization across several different feature types.

3.1. Hard bins

Our first approach for parameterizing these distributions was to simply discretize each feature separately and then count the number of times each feature fell in each bin during the utterance [1]. Since we do not know a priori where to place thresholds for binning the data, we discretize evenly on the rank distribution of values for the particular feature, so that resulting bins contain roughly equal amounts of data. When this is not possible, as in the case of discrete features, we allow unequal mass bins. For pauses we use one set of hand-chosen threshold values (60, 150, and 300 ms) to divide them into four different lengths. In this approach undefined value are simply grouped into a separate bin. The bins for bigrams and trigrams are obtained by concatenating the bins for each feature in the sequence. This results in a grid and the features are simply the counts corresponding to each bin in the grid. In all cases the counts are normalized by the total number of syllables in the sample. Many of the bins obtained by simple concatenation will correspond to places in the feature space where very few samples ever fall. The vector $b(X)$ is then composed only of the counts corresponding to bins for which the count was higher than a certain threshold in some held-out data.

This approach has a few weaknesses. First, the method generates inherently noisy features since small variations in the raw value of the feature may result in the feature falling in two different bins. In our experiments this effect is compensated for in part by the use of two versions of binnings for each N-gram length, one with lower resolution than the other. Second, since the bins for bigrams and trigrams are obtained by simple concatenation of the bins for unigrams, we have no guarantee that we are covering the feature space in any optimal way. Finally, the algorithm to compute the bin thresholds requires heuristics to deal with the continuous/discrete nature of these features. This, along with the need for pruning the list of possible bins to get rid of the unfrequent ones, makes the approach very inelegant. The next two approaches aim to deal with these problems.

3.2. GMM soft bins using EM

The main idea behind this method is to model each token with a Gaussian Mixture Model (GMM) and use the weights of the gaussians to form the vector $b(X)$. The procedure is as follows: A GMM is trained using the EM algorithm (initialized using the VQ algorithm described in Section 3.3 to ensure a good starting point) for each token using the pooled data from a few thousand speakers. The vectors used to train the GMM for a token corresponding to feature f_j and pattern $Q = (q_0, ...q_{N-1})$ (where q_i is either P for pause, or S for syllable) are of the form $Y_j^{(t)} = (y_{j,0}^{(t)}, ...y_{j,N-1}^{(t)})$ where t is the slot index (from 1 to T) and

$$y_{j,k}^{(t)} = \begin{cases} log(p^{(t+k)}) & \text{if } q_k = P \\ f_j^{(t+k)} & \text{if } k = 0 \text{ or } q_{k-1} = P \\ f_j^{(t+k)} - f_j^{(t+k-1)} & \text{otherwise,} \end{cases} \tag{1}$$

where $p^{(t)}$ is the length of the pause at slot t and $f_j^{(t)}$ is the value of prosodic feature f_j at slot t. The logarithm is used to reflect the fact that the influence of the length of the pause decreases as the length increases. In this approach the undefined values are treated exactly as in Method 2 in [6], i.e., a bootstrap model is first trained using only vectors for which all features are defined, then the parameters are reestimated using all data (except vectors that are completely undefined). See [6] for details on the implementation. Discrete features are treated in the same way as continuous ones, with the only precaution that variances that become too small are clipped to a minimum value. This seems to work reasonably well except for cases in which the distribution has one single value much more likely than others. In this case EM tends to accumulate most gaussians on the same value, leaving the least likely values unattended.

Once the background GMMs for each token have been trained, the features for each test and train sample are obtained by maximum a posteriori (MAP) adaptation of the GMM weights to the sample's data. The adapted weights are simply the posterior probabilities of the Gaussians given the feature vector, averaged over all syllables in the utterance. The adapted weights for all tokens are finally concatenated to form $b(X)$. A simple smoothing procedure where the adapted weight is obtained as a convex combination of the weight in the original GMM and the average of the posteriors for the utterance was tried but did not show a consistent improvement across different databases and conditions.

For the unidimensional case (unigrams), the procedure described is closely related to the hard bin method described above with the bins replaced by gaussians and the counts by posteriors. For longer N-grams, there is a bigger difference: the soft bins represented by the gaussians are obtained by looking at the joint distribution from all dimensions, while in the previous method, the bins were obtained as a concatenation of the bins for the unigrams.

This method is similar to the one presented in [5] in that a GMM is trained (using the EM algorithm) on data from many different speakers and MAP adapted to each speaker's data. The two main differences between the two methods are, first, that in our case there is not a single background GMM model, but rather one per token, second, that we use the weights as parameters rather than the means and variances. This approach was chosen as a natural transition from the hard-bins approach. We believe that for this kind of features, which are sparser than cepstral features and higher dimensional, the single background GMM may not be a good approach since there would not be enough samples in the speaker's data to perform the adaptation to such a high-dimensional model (unless the number of gaussians is very small, in which case the power of the model may be restricted).

3.3. GMM soft bins using VQ

The GMMs obtained using EM are designed to maximize the likelihood of the data given the model. The result is usually a model where the gaussians overlap each other in order to approximate the actual distribution of the data as well as possible. This, in turn, results in features (weights corresponding to those gaussians) that are highly correlated with each other and that may have limited power to discriminate between small changes in small regions of the space. Furthermore, as mentioned before, the models resulting from EM when discrete values are involved are less than ideal. To overcome these problems, we decided to look into vector quantization (VQ) as a different approach for training the background GMMs. The vectors used in this approach are defined as in the previous method, by Equation 1, and the final features for each sample are obtained by doing MAP adaptation of the background GMMs to the sample's data, also as above.

A variation of the LBG algorithm is used to create the models [7]. Initially, the Lloyd algorithm is used to create two clusters. The cluster with higher total distortion is then further split in two by perturbing the mean of the original cluster by a small amount. These clusters are used as a starting point for running a few iterations of the Lloyd algorithm. The algorithm continues splitting one cluster at a time until the desired number of clusters is reached. During every step the distortion used is weighted squares, i.e., $d(x, y) = \sum (x_i - y_i)^2/v_i$, where v_i is the global variance of the data in the dimension i. When an undefined feature is present, the term corresponding to that dimension is simply ignored in the computation of the distortion. If at any step a cluster is created that has too few samples, this cluster is destroyed and a cluster with high total distortion is split in two. Once the final set of clusters is obtained, a GMM is created by assigning one gaussian to each cluster with mean and variance determined by the data in the cluster and weight given by the proportion of samples in that cluster. This approach naturally deals with discrete values resulting in clusters with a single discrete value when necessary. The variances for these clusters are set to a minimum when converting the codebook to a GMM. Figure 1 shows a comparison of the gaussians obtained using EM and VQ for the same feature. Intuitively, it seems that the model obtained with VQ should be able to describe more of the details in the distribution, but it would also be more sensitive to sparse data, since each weight corresponds to a smaller range in the feature space. As we will see in the next sections the results seem to support this intuition.

4. EXPERIMENTS

Experiments were conducted using data from the NIST SRE from 2005 and 2006. Each speaker verification trial consists of a test sample and a speaker model. The test samples are one side of a telephone conversation with approximately 2.5 minutes of speech. We consider the 1-side and 8-side training conditions in which we are given 1 or 8 conversation sides to train the speaker model. Each of these conversations corresponds to one positive example when training the SVM model for the speaker. The data used as negative examples for the SVM training is taken from 2003 and 2004 NIST evaluations along with some FISHER data, resulting in a total of 2122 conversation sides. These same samples are used to compute the distribution of each feature to perform the rank-normalization. The SRE2005 and SRE2006 tasks contain 25,887 and 24,004 trials for the 1-side training condition and 17,216 and 15,105 trials for the 8-side training condition. The average number of syllables per conversation side is around 600. Trials involving a test or train conversation side with fewer than 60 syllables where removed from the original list prepared by NIST resulting on the number of trials mentioned above.

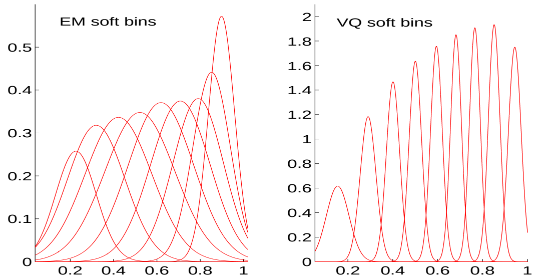

Fig. 1. Comparison of gaussians generated by EM and VQ

The syllable-level features for each test and train conversation sides (positive and negative examples) are transformed into conversation side-level vectors using the three methods described above. The data used to obtain the equal mass bins in the first method and to obtain the background models for the tokens in the other two methods was drawn from data from the 2003 and 2004 NIST evaluations along with some FISHER data, yielding a total of 2456 conversation sides from 1228 unique speakers (2 sides per speaker) with little overlap with the negative example data.

Figure 2 shows a comparison of results for the three methods on SRE2005 data for both training conditions for four different systems: three that use unigrams, bigrams, and trigrams only and one that uses all three sets. The performance measure presented in Figure 2 is equal error rate (EER), the false acceptance rate obtained when the score threshold is tuned to achieve an equal number of false acceptances and false rejections. Parameters were chosen so that all three methods result in approximately the same number of features for each system: 11,000 for unigrams, 13,000 for bigrams, 14,000 for trigrams, and 38,000 for the complete system.

In the case of hard bins this number of features is reached by using the most frequent bins from two different resolutions, a finer one (of 72, 16 and 8 for unigrams, bigrams and trigrams) and a coarser one of approximately 1/4th the resolution of the finer one. For both soft bin methods, the number of components of each GMM is determined as $N_g = \frac{N_s}{S}$ where N_s is the average number of samples per conversation side available for training the model and S is a tunable parameter. For the EM method, setting S=6.4 in the formula for N_g results in approximately the right number of features for all N-gram lengths. For the VQ method, two resolutions are used, one with $S = 8$ and one with $S = 35$ (i.e., a finer one and a coarser one of around 1/4th the resolution, as for the hard bins) which, after merging, result in the right number of features. For the first and third methods, the use of a finer along with a coarser bin resolution resulted in significant gains in performance. On the other hand, for the second method a single fine resolution proved to be the best choice. This is reasonable given that EM results in gaussians that usually span greater ranges of the space than either of the other two methods, resulting in more robust features, avoiding the need for a coarser resolution.

In Figure 2 we see that for all three systems, bigrams and trigrams, which model the development of the features over time, perform at least 20% better than the unigrams which simply model the distribution of the features. The gain is bigger when more training data is available. For all N-gram lengths and for the complete system, both soft bin systems outperform hard bins significantly [1] except for the case of unigrams and bigrams for 8-side training, where

[1] Significance is determined with a McNemar test at level 0.05

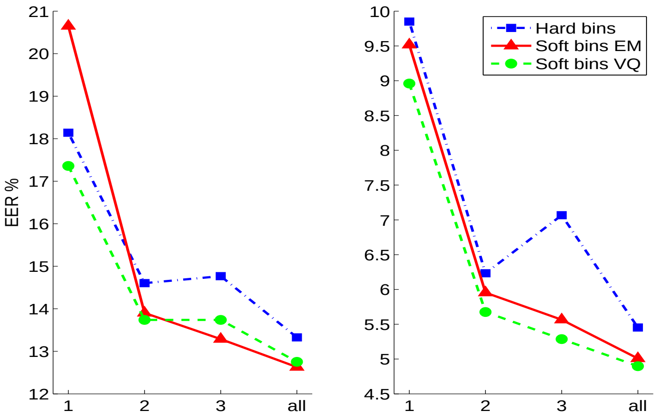

Fig. 2. EER results for the three methods on SRE2005 data 1-side training (left) and 8-side training (right) for four different systems: using unigrams, bigrams and trigrams only and using all three sets.

EM is not significantly better than hard bins and for the case of unigrams with 1-side training, where EM is significantly worse than the other two methods. When comparing both soft bin approaches we see that VQ features are significantly better than EM features for the unigram case on both training conditions. In all other cases, VQ and EM are not significantly different except for the case of trigrams on 1-side training condition where EM is significantly better than VQ. Our interpretation of these results is that EM features cannot capture the small differences between speakers as well as the other two sets of features where there is no or little overlap between the bins, allowing for faster adaptation to the speaker's distribution. When more data is available, this effect is diminished. Furthermore, the better treatment of discrete values gives an advantage to VQ features. Both effects are balanced out by the robustness of the EM features for higher dimensions and/or less training data.

SRE2006 was used as test set since all development was performed on SRE2005 data. Table 1 shows results on this database for each of the three SNERF systems alone (using the complete systems with the three N-gram lengths included) and in combination with a state-of-the-art MLLR-SVM system [8]. This system was chosen for combination since it is currently the best of our three cepstral-based systems. T-Norm is applied on the MLLR system, but not on the SNERF systems. The table shows minimum detection cost function (DCF) values along with EER. DCF is the main criterion in the NIST SRE and is defined as the Bayesian risk with P_{target}=0.01, C_{fa}=1, and C_{miss}=10. The combination was performed using a single layer neural network trained on SRE05 data.

Results on SRE06 maintain the improvement when soft bins are used as opposed to hard bins, although the differences in EER in 8-side training are not significant. As noticed before, the VQ soft bins seem to lead to less robust features than the EM soft bins, making EM a better choice for SRE06 data which has been repeatedly shown to be considerably different from SRE05 and SRE04 data. When combining the EM SNERF system with the MLLR system we see significant improvements in the EER of 8% for 1-side training and 19% for 8-side training. Furthermore, these results are significantly better than the ones obtained when combining with the hard bin SNERF system.

5. CONCLUSIONS

We have presented three approaches to the problem of transforming a set of prosodic features into a single sample-level vector that can be input to an SVM, for high-performance speaker recognition. The transformations model both individual feature distributions and the pattern of feature values over time. The methods proposed in this paper can potentially be applied in any binary classification task where features with mixed continuous, discrete and undefined values are used.

System	1-side		8-sides	
	EER	DCF	EER	DCF
Hard bins SNERFs	14.19	0.6087	5.19	0.2572
VQ soft bins SNERFs	13.17	0.5732	4.91	0.2285
EM soft bins SNERFs	12.30	0.5538	4.85	0.2247
MLLR only	4.59	0.2125	2.05	0.0828
EM SNERFs + MLLR	4.21	0.1953	1.65	0.0718

Table 1. Results on SRE06 for the three different SNERF systems alone and for the best of them (EM SNERFs) in combination with the MLLR system.

Results show clearly that the two methods that use a soft representation of the feature bins, in the form of a GMM, outperform the simpler method that uses hard bins. Furthermore, training the GMM with the VQ algorithm gives a clear advantage for feature vectors of dimension one, while for higher dimensions, both methods perform similarly. When the resulting prosodic system employing EM is combined with our best-performing baseline acoustic system (an MLLR system), the result is an 8% EER reduction over the MLLR system alone for the 1-side training condition and a 19% EER reduction for 8-side training.

6. ACKNOWLEDGMENTS

This work was funded by NSF IIS-0544682. The views herein are those of the authors and do not reflect the views of the funding agencies. We thank the compression group at Stanford for fruitful discussions.

7. REFERENCES

[1] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke, "Modeling prosodic feature sequences for speaker recognition," *Speech Communication*, vol. 46, no. 3-4, pp. 455–472, 2005, Special Issue on Quantitative Prosody Modelling for Natural Speech Description and Generation.

[2] L. Ferrer, E. Shriberg, S. Kajarekar, A. Stolcke, K. Sönmez, A. Venkataraman, and H. Bratt, "The contribution of cepstral and stylistic features to SRI's 2005 NIST speaker recognition evaluation system," in *Proc. ICASSP*, Toulouse, May 2006, vol. 1, pp. 101–104.

[3] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," in *Proc. ICASSP*, Orlando, FL, May 2002, vol. 1, pp. 161–164.

[4] V. Wan and S. Renals, "Speaker verification using sequence discriminant support vector machines," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 2, 2005.

[5] W. M. Campbell, D. Sturim, and D. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Processing Letters*, vol. 13, no. 5, May 2006.

[6] S. Kajarekar, L. Ferrer, K. Sonmez, J. Zheng, E. Shriberg, and A. Stolcke, "Modeling NERFs for speaker recognition," in *Proc. Odyssey-04 Speaker and Language Recognition Workshop*, Toledo, Spain, May 2004, pp. 51–56.

[7] Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers Group, Norwell, Massachusetts, 1992.

[8] A. Stolcke, L. Ferrer, and S. Kajarekar, "Improvements in MLLR-transform-based speaker recognition," in *Proc. IEEE Odyssey 2006 Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, June 2006.