POLYNOMIAL CLASSIFIER TECHNIQUES FOR SPEAKER VERIFICATION

W. M. Campbell

Motorola SSG
Scottsdale, AZ 85257, USA

K. T. Assaleh

Rockwell Semiconductor Systems
Newport Beach, CA 92660, USA

ABSTRACT

Modern speaker verification applications require high accuracy at low complexity. We propose the use of a polynomial-based classifier to achieve this objective. We demonstrate a new combination of techniques which makes polynomial classification accurate and powerful for speaker verification. We show that discriminative training of polynomial classifiers can be performed on large data sets. A prior probability compensation method is detailed which increases accuracy and normalizes the output score range. Results are given for the application of the new methods to YOHO.

1. INTRODUCTION

Recently, speaker verification has become an attractive biometric for implementation in computer networks. Voice verification has the advantage that it requires little custom hardware and is non-intrusive. One of the needs of current systems is to obtain reasonable accuracy at low computational complexity.

Many structures have been proposed for speaker verification. The two most popular techniques are statistical and discriminative-training based methods. For the former, Gaussian Mixture Models (GMM) [1] and HMM systems with cohort normalization [2, 3] have been the techniques of choice. For discriminative systems, neural tree networks and multilayer perceptrons [4] have been commonly used. In this paper, we propose a new text-independent speaker verification system based upon a discriminatively-trained polynomial-based classifier.

Polynomial classifiers have been known in the literature for many years [5, 6]. Polynomials have excellent properties as classifiers. Because of the Weierstrass theorem, polynomial classifiers are universal approximators to the optimal Bayes classifier [7].

Typical methods of training polynomial classifiers have either been based on statistical methods or minimizing a mean-squared error criterion. The focus has been on linear or second order classifiers. Both of these methods traditionally have had limitations. Statistical methods estimate the mean and covariance of the speaker data for a parametric model. The drawback of this approach is that out-of-class data is not used to maximize performance. Discriminative methods usually involve large matrices which leads to large intractable problems.

We propose an approach which makes the discriminative training separable. This approach makes polynomial training tractable for large data sets.

Section 2 describes the structure of the polynomial classifier. Section 3 illustrates the new training algorithm. Section 4 shows how a simple method for prior compensation can be integrated into the training method. Section 5 applies the method to the YOHO database.

2. CLASSIFIER STRUCTURE

The basic scenario for verification is as follows. An identity claim is made by a speaker (the claimant). The model for the claimant is then retrieved. Speech is obtained from claimant. Feature extraction is performed on the speech. The features are passed through a classifier to produce a score. A decision is then made based upon whether the score is above or below a threshold.

The basic embodiment of this strategy in polynomial form is shown in Figure 1. The classifier consists of several parts. Feature vectors, $\mathbf{x}_1, \dots, \mathbf{x}_N$ are introduced into the classifier. The vector $p(\mathbf{x})$ is the vector of polynomial basis terms of the input feature vector; e.g., for two features, $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^t$ and second order, the vector is given by

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

In general, the polynomial basis terms of the form

$$x_{i_1} x_{i_2} \dots x_{i_k} \tag{2}$$

are used where k is less than or equal to the polynomial order, K. The speaker model is given by $\mathbf{w}$. For each input feature vector, $\mathbf{x}_i$, a score is produced by the inner product, $\mathbf{w}^t p(\mathbf{x}_i)$. The score is then averaged over time to produce

The work was supported by Motorola SSG under internal research and development funding. The work of the second author was done while he was at Motorola SSG.

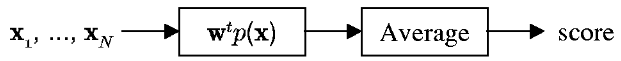
$\mathbf{x}_1, \ldots, \mathbf{x}_N$
$\mathbf{w}^t p(\mathbf{x})$
Average
score

Figure 1: Polynomial classifier for verification.

the final output. The total score is then

$$s = \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}^t p(\mathbf{x}_i). \qquad (3)$$

Viewed in another manner, we are averaging the output of a discriminant function [5] over time.

The accept/reject decision for the system in Figure 1 is performed by comparing the output score (3) to a threshold. If $s < T$, then reject the claim; otherwise, accept the claim.

3. TRAINING METHOD

In order to obtain good separation between classes for the system in Figure 1, we use discriminative training with a mean-squared error criterion. For the speaker's features an output of 1 is desired. For impostor data, an output of 0 is desired. The resulting problem is

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{N_{\text{spk}}} \left| \mathbf{w}^t p(\mathbf{x}_i) - 1 \right|^2 + \sum_{i=1}^{N_{\text{imp}}} \left| \mathbf{w}^t p(\mathbf{y}_i) \right|^2 \right]. \qquad (4)$$

Here, the speaker's training data consists of $\mathbf{x}_1, \ldots, \mathbf{x}_{N_{\text{spk}}}$, and the example impostor data consists of $\mathbf{y}_1, \ldots, \mathbf{y}_{N_{\text{imp}}}$.

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

$$\mathbf{M}_{\text{spk}} = \begin{bmatrix} p(\mathbf{x}_1)^t \\ p(\mathbf{x}_2)^t \\ \vdots \\ p(\mathbf{x}_{N_{\text{spk}}})^t \end{bmatrix}. \qquad (5)$$

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

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{\text{spk}} \\ \mathbf{M}_{\text{imp}} \end{bmatrix}. \qquad (6)$$

The problem (4) then becomes

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{M}\mathbf{w} - \mathbf{o}\|_2 \qquad (7)$$

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

The problem (7) can be solved using the method of normal equations [8],

$$\mathbf{M}^t\mathbf{M}\mathbf{w} = \mathbf{M}^t\mathbf{o}. \qquad (8)$$

We rearrange (8) to

$$\left(\mathbf{M}_{\text{spk}}^t \mathbf{M}_{\text{spk}} + \mathbf{M}_{\text{imp}}^t \mathbf{M}_{\text{imp}}\right) \mathbf{w} = \mathbf{M}_{\text{spk}}^t \mathbf{1} \qquad (9)$$

where $\mathbf{1}$ is the vector of all ones. If we define $\mathbf{R}_{\text{spk}} = \mathbf{M}_{\text{spk}}^t \mathbf{M}_{\text{spk}}$ and define $\mathbf{R}_{\text{imp}}$ similarly, then (9) becomes

$$\left(\mathbf{R}_{\text{spk}} + \mathbf{R}_{\text{imp}}\right) \mathbf{w} = \mathbf{M}_{\text{spk}}^t \mathbf{1}. \qquad (10)$$

Our new training method is based on (10). There are several advantages to this approach. First, the matrices $\mathbf{R}_{\text{imp}}$ and $\mathbf{R}_{\text{spk}}$ are fixed in size; i.e., they do not vary with the size of the training data set. Let M_{terms} equal the number of terms in $p(\mathbf{x})$, then $\mathbf{R}_{\text{imp}}$ and $\mathbf{R}_{\text{spk}}$ are matrices of size $M_{\text{terms}} \times M_{\text{terms}}$. Second, the computation is partitioned. We can calculate $\mathbf{R}_{\text{spk}}$ and $\mathbf{R}_{\text{imp}}$ separately at costs of $O(N_{\text{spk}} {M_{\text{terms}}}^2)$ and $O(N_{\text{imp}} {M_{\text{terms}}}^2)$, respectively. The calculation of these two matrices is the most significant part of computation. Note that $\mathbf{R}_{\text{imp}}$ can be precomputed and stored. Third, the terms in the right-hand side of (10) can be calculated as a submatrix of $\mathbf{R}_{\text{spk}}$. We denote the resulting vector, $\mathbf{a}_{\text{spk}}$.

One potential drawback of the method should be noted. Since the normal equation method squares the condition number of the matrix $\mathbf{M}$, it might be thought there would be problems solving (9). In practice, this squaring has not caused solution instability. Many linear approximation problems have large condition numbers (e.g., linear FIR filter design), but yield good results.

The matrix $\mathbf{R}_{\text{spk}}$ (and its impostor counterpart) in (10) has many redundant terms. In order to reduce storage and computation, only the unique terms should be calculated. The terms in $\mathbf{R}_{\text{spk}}$ consist exactly of sums of the polynomial basis terms of order $\leq 2K$. We denote the terms of order $\leq 2K$ as $p_2(\mathbf{x})$. Table 1 shows the number of unique terms in $\mathbf{R}_{\text{spk}}$ for 12 features and different polynomial orders. The resulting training algorithm is shown in Table 2.

Table 1: Term Redundancies for the Matrix $\mathbf{R}_{\text{spk}}$.

Order	Terms in $\mathbf{R}_{\text{spk}}$	Unique Terms	Ratio
2	8,281	1,820	4.55
3	207,025	18,564	11.15
4	3,312,400	125,970	26.30

The redundancy in $\mathbf{R}_{\text{spk}}$ is very structured. Suppose we have the polynomial basis terms of order k, and we wish to calculate the terms of order $k+1$. Assume that every term is of the form (2) where $i_1 \leq i_2 \leq \cdots \leq i_k$. If we have the kth order terms with end term having $i_k = l$ as a vector $\mathbf{u}_l$, we can obtain the $(k+1)$-th order terms ending with

Table 2: Training Algorithm.

```
1) Let $\mathbf{r}_{\text{imp}} = \mathbf{0}$ and $i = 1$.
2) Retrieve feature vector $\mathbf{y}_i$ from background set of impostors.
3) Let $\mathbf{r}_{\text{imp}} = \mathbf{r}_{\text{imp}} + p_2(\mathbf{y}_i)$.
4) Let $i = i + 1$. Goto step 2 until all impostor feature vectors are processed.
5) Repeat steps 6 to 13 for each speaker $k$.
6) Set $\mathbf{r}_{\text{spk},k} = \mathbf{0}$ and $i = 1$.
7) Retrieve feature vector $i$, $\mathbf{x}_{k,i}$, for speaker $k$.
8) Let $\mathbf{r}_{\text{spk},k} = \mathbf{r}_{\text{spk},k} + p_2(\mathbf{x}_{k,i})$.
9) Let $i = i + 1$. Go to step 7 until all of speaker $k$'s feature vectors are processed.
10) Map $\mathbf{r}_{\text{spk},k}$ to $\mathbf{R}_{\text{spk}}$, $\mathbf{r}_{\text{spk},k}$ to $\mathbf{a}_{\text{spk},k}$, and $\mathbf{r}_{\text{imp}}$ to $\mathbf{R}_{\text{imp}}$.
11) Compute $\mathbf{R} = \mathbf{R}_{\text{spk}} + \mathbf{R}_{\text{imp}}$.
12) Compute the Cholesky decomposition of $\mathbf{R}$, $\mathbf{R} = \mathbf{L}^t\mathbf{L}$.
13) Solve for $\mathbf{w}_{\text{spk},k}$ by using a triangular solver twice on $\mathbf{L}^t\mathbf{L}\mathbf{w}_{\text{spk},k} = \mathbf{a}_{\text{spk},k}$.
```

$i_{k+1} = l$ as

$$\begin{bmatrix} x_l\mathbf{u}_1 \\ x_l\mathbf{u}_2 \\ \vdots \\ x_l\mathbf{u}_l \end{bmatrix}. \tag{11}$$

We can then construct $p(\mathbf{x})$ as follows. Initialize a vector of 1 and the first order terms. Then recursively calculate the $(k+1)$-th order terms from the kth using (11). Concatenate the different orders to get the final result. One can then plot the index of terms in the vector $p_2(\mathbf{x})$versus the index of the terms in the matrix $\mathbf{R}_{\text{spk}}$. We index the elements in $\mathbf{R}_{\text{spk}}$ as a one-dimensional array using column major form. The resulting plot is shown in Figure 2 for 8 features and a polynomial order of 3. Note the interesting self-similar structure. This structure becomes more detailed as the polynomial order is increased.

4. PRIOR COMPENSATION

The prior probabilities of the speaker training set and impostor training set are usually imbalanced because much more impostor data is available. There are several approaches to this problem including balancing the amount of data that the training algorithm sees per epoch and division of the classifier output by priors, see [9]. We propose a different approach incorporated into the training algorithm.

In order to compensate for the prior probabilities, we weight the criterion in (4) as follows:

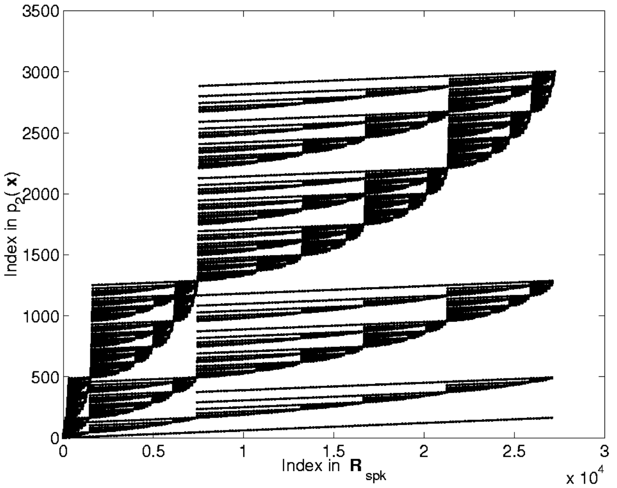

Figure 2: Plot of index of term in $p_2(\mathbf{x})$ versus index in $\mathbf{R}_{\text{spk}}$.

$$\begin{aligned} \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \Bigg[& \frac{1}{N_{\text{spk}}} \sum_{i=1}^{N_{\text{spk}}} \left|\mathbf{w}^t p(\mathbf{x}_i) - 1\right|^2 \\ & + \frac{1}{N_{\text{imp}}} \sum_{i=1}^{N_{\text{imp}}} \left|\mathbf{w}^t p(\mathbf{y}_i)\right|^2 \Bigg]. \end{aligned} \tag{12}$$

The effect of this weighting is twofold. First, overtraining is reduced. The contribution to the total mean-squared error is equalized out in (12). Second, the score range is normalized. This normalization has the effect that as the amount of impostor data is increased, the score range stays approximately the same.

The prior compensation can be easily incorporated into the training algorithm. The new problem becomes

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \|\mathbf{DMw} - \mathbf{Do}\|_2 \tag{13}$$

where $\mathbf{D}$ is a diagonal matrix. The method in Table 2 is easily adapted to the new training equation (13).

5. RESULTS

We applied our method to the YOHO speaker verification database [10, 11]. YOHO uses prompted combination lock phrases, e.g., "26-81-57." YOHO has four enrollment sessions with 24 phrases per session. Verification consists of 10 sessions with 4 phrases per session.

Feature analysis was performed by using 12th order LP analysis on a 30 millisecond frame every 10 milliseconds. Each frame was pre-emphasized with the filter $1 - 0.97z^{-1}$, and a Hamming window applied to the result. From the LP coefficients, 12 LP cepstral coefficients (LPCC) and delta-cepstral coefficients were produced.

For enrollment, all 4 sessions of YOHO were used. For one-phrase verification, all 40 utterances per speaker were used. For four-phrase verification, we grouped all phrases within a session. The classifier in Figure 1 was used in text independent mode. That is, one model was generated per speaker without using knowledge of the particular text spoken.

Table 3 shows a comparison of the average equal error rate (EER) for the classifier at different orders. Prior compensation, see Section 4, and the algorithm in Table 2 were used. The average EER was computed by finding the EER for each speaker and then averaging across all speakers. Note that the performance of the new technique compares favorably to other systems [11]. The best compromise between accuracy, computation, and storage appears to be the 3rd order system which has 455 model parameters and 18564 entries in $p_2(\mathbf{x})$.

Table 3: Classifier performance on the YOHO database.

cep	dcep	order	Avg. EER % 1 phrase	Avg. EER % 4 phrase
12	-	2	3.62	0.90
12	-	3	1.52	0.28
12	-	4	1.00	0.14
12	12	2	1.81	0.32

To illustrate the effect of prior compensation on accuracy, we trained without the weighting in (13). For the 4 phrase test, 12 cepstral coefficients, and 3rd order, the resulting average EER was 0.37%; i.e., a decrease in accuracy over the weighted case was observed. The prior compensation normalized the score range, so that typical EER thresholds were around 0.5.

One concern about the YOHO database is the lack of a background separate from the verification population. Our original enrollment and verification strategy used all 138 speakers for training and testing. We also tried enrolling the first 69 speakers and second 69 speakers separately. Speakers 1 through 69 were trained only against 1 to 69; speakers 70 through 138 were trained only against speakers 70 through 138. Verification was performed by using the second 69 speakers as impostors for the first 69 and vice versa. The resulting 1 phrase EER for 12 cepstral coefficients and 3rd order was 1.63%. The result is close to the one shown in Table 3; this test shows that the polynomial classifier works well with an unknown impostor set. The comparison between the two training approaches is not entirely valid since a larger training background and a larger number of impostors is available for the case in Table 3.

6. SUMMARY AND CONCLUSIONS

A new method of applying a polynomial classifier to speaker verification was shown. Algorithms were presented to deal with large amounts of data as well as efficiently train speaker verification models. These techniques were applied to the YOHO database showing that the resulting system achieved excellent verification performance at low complexity.

7. REFERENCES

[1] D. A. Reynolds, "Automatic speaker recognition using Gaussian mixture speaker models," *The Lincoln Laboratory Journal*, vol. 8, no. 2, pp. 173–192, 1995.

[2] A. Higgins, L. Bahler, and J. Porter, "Speaker verification using randomized phrase prompting," *Digital Signal Processing*, vol. 1, pp. 89–106, 1991.

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

[4] K. R. Farrell, R. J. Mammone, and K. T. Assaleh, "Speaker recognition using neural networks and conventional classifiers," *IEEE Trans. on Speech and Audio Processing*, vol. 2, pp. 194–205, Jan. 1994.

[5] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[6] D. F. Specht, "Generation of polynomial discriminant functions for pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-16, pp. 308–319, June 1967.

[7] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*. John Hopkins, 1989.

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

[10] J. P. Campbell, Jr., "Testing with the YOHO CD-ROM voice verification corpus," in *Proceedings of the Internation Conference on Acoustics, Speech, and Signal Processing*, pp. 341–344, 1995.

[11] J. P. Campbell, Jr., "Speaker recognition: A tutorial," *Proceedings of the IEEE*, vol. 85, pp. 1437–1462, Sept. 1997.