Neural, Parallel, and Scientific Computations 17 (2009) 215 – 238

Speaker Recognition Using Parallel Neural Network Modules

Glen E. Deal[1] and Fredric M. Ham[2]

[1] Northrop Grumman Corporation,
Aerospace Systems Sector, Melbourne, Florida, USA
[2] Department of Electrical and Computer Engineering,
Florida Institute of Technology, Melbourne, FL, USA

Abstract

This paper presents methods for improving the performance of Artificial Neural Network Speaker Recognition systems based on a novel negative reinforcement training algorithm. A brief background of current speaker recognition technology is presented including various options for feature extraction and feature mapping. We then present custom MATLAB implementations for four separate closed-set speaker recognition systems, each using a different parallel-architecture neural network mapping algorithm. The performance of the negative reinforcement training algorithm as applied to each of the four mapping algorithms is tested and illustrated a set of Receiver Operating Curves. The negative reinforcement training algorithm is shown to provide a significant improvement in correct recognition rate over a similarly configured baseline system.

1. INTRODUCTION

The problem of speaker recognition generally consists of mapping a sample of speech to a particular speaker based on a set of speech parameter measurements or "feature vectors". Possible applications of speaker recognition technology include speaker verification for access to a physical site or to a computer network, telephone banking transaction authentication, and security screening for airline passengers, to name a few. This problem has been the subject of much recent research [1-19]. The problem is usually broken into two distinct phases: feature extraction and feature mapping.

Feature extraction is concerned with producing a numeric representation of a speech sample that can be processed by a computer. A very simple feature extraction method could be implemented by taking the Fast Fourier Transform (FFT) of the speech sample and using the FFT coefficients as the feature vector. More sophisticated algorithms include Principal Component Analysis, Wavelet Analysis, and Cepstral Vector Analysis. The main goal of the feature extraction algorithm is to produce a representation that is dependent only on the speaker and independent of the content of the speech. In this

Received February 13, 2009 1061-5369 $15.00 © Dynamic Publishers, Inc.

respect, speaker recognition has the opposite problem from speech recognition, where one wants the feature vector to be dependent only on the content of the speech and independent of the speaker. The recognizers developed and demonstrated as part of this paper use two options for feature extraction: Cepstral Vector feature extraction and Wavelet feature extraction. The wavelet feature vectors are produced using the Princeton WAVELAB_802 software. The algorithm for extracting the cepstral vectors for a speech sample has been implemented in custom MATLAB code and will be presented in detail in Section 2.

Once a feature vector is obtained, the vector must be mapped to a speaker. If the speaker is known to be one of a set of N possible speakers, the problem is categorized as a "closed-set" recognition problem. If the formulation of the problem allows the speaker to be outside the set of known speakers, we have an "open-set" recognition problem. In either case, there are a number of options for algorithms to map the feature vector to a speaker. The classical Bayesian approach is known to have the best performance from a probability of error standpoint if the marginal probability distributions of each feature vector component with respect to the members of the speaker set are known [20]. Other options include Principal Component Regression techniques and Nearest Neighbor methods. In practice, methods based on parallel architecture Artificial Neural Networks (ANNs) have been shown to have superior performance [2]. The four recognizers developed and demonstrated here all use parallel ANN schemes in which a separate ANN is trained for each speaker in the speaker set. The four separate feature mapping algorithms along with the improved negative reinforcement training approach are described in detail in Sections 3-8.

2. FEATURE EXTRACTION

Our approach involves implementing a baseline recognition system using the best current algorithms for speaker recognition and then demonstrating a significant improvement in performance over the baseline system using the negative reinforcement algorithm. The baseline system uses a 15-element Cepstral Vector Feature Extraction algorithm and then implements an Artificial Neural Network to map the feature vectors to members of a closed speaker set.

2.1 Cepstral Vector Feature Extraction

As discussed in Section 1, the main goal of the feature extraction algorithm in a speaker recognition system is to produce a numeric representation of a speech sample that is characteristic of the speaker and independent of the content of the speech. Cepstral Vector Feature Extraction has been shown to have superior performance in this respect [3]. The following steps describe the process for extracting a feature vector for each speech sample [21].

Step 1: Normalize the discrete-time speech sample and remove the mean:
For a discrete-time speech sample y = y(k), k=1,N

$$\mathbf{y}_{\text{normalized}} = \mathbf{y} / \max(|\mathbf{y}|) \tag{1}$$

$$\mathbf{y}_{\text{n, m}} = \mathbf{y}_{\text{normalized, mean removed}} = \mathbf{y}_{\text{normalized}} - \text{mean}(\mathbf{y}_{\text{normalized}}) \tag{2}$$

Step 2: Apply a symmetric Hamming Window:

$$\mathbf{y}_{\text{n, m, h}} = \mathbf{y}_{\text{n, m}} \cdot [\mathbf{W}_{\text{hamming}} \mid \mathbf{W}^{*}_{\text{hamming}}] \tag{3}$$

$$\mathbf{W}_{hamming} = \frac{54 - 46 \times \cos(\frac{2\pi h}{N-1})}{100}, \quad where \quad h = 0,1,\ldots\frac{N}{2} - 1. \tag{4}$$

Step 3: Calculate the Power Spectral Density $S_{yy}(k)$ of the signal $\mathbf{y}_{n,m,h}$

$$S_{yy}(\omega) = \sum_{k=-\infty}^{\infty} R_{yy}(k) e^{-j\omega k} \tag{5}$$

where $R_{yy}(k)$ is the autocorrelation of the signal $\mathbf{y}_{n,m,h}$ (estimated as the periodogram of the signal)

Step 4: Apply Mel-frequency scaling to the PSD S(k):

$$S_m(k) = \alpha \log_e[\beta S(k)]$$
$$\text{where } \alpha = 1125, \quad \beta = 0.000003 \tag{6}$$

Step 5: Take the inverse discrete cosine transform:

$$x_m(n) = \frac{1}{n} \sum_{k=0}^{N-1} S_m(k) \cos(2\pi kn / N), \text{ for n = 0, 1, 2 …, N-1.} \tag{7}$$

Step 6: Take the derivative of the sequence $x_m(n)$, i.e., $x'_m(n)$

Step 7: Concatenate $x'_m(n)$ with $x_m(n)$ to form the augmented sequence:

$$x_m^a = [x'_m(i) \mid x_m(j)]$$
$$\text{where i = 1, 2, …5 and j = 1, 2, …10} \tag{8}$$

Step 8: Take the absolute value of x_m^a:

$$x_{m,abs}^a = \left| x_m^a \right| \tag{9}$$

Step 9: Take the natural log of $x_{m,abs}^a$:

$$x_{m,abs,\log}^a = \log_e [x_{m,abs}^a] \tag{10}$$

Step 10: Remove the mean value:

$$x_{meanremoved} = x_{m,abs,\log}^a - mean(x_{m,abs,\log}^a) \tag{11}$$

Step 11: Scale the range of the feature vector between [-1, 1]:

$$x_{mean\,removed,normalized} = x_{mean\,removed} / \max(\left| x_{mean\,removed} \right|) \tag{12}$$

2.2 Wavelet Feature Extraction

In addition to the Cepstral Vector Feature Extraction algorithm (which was implemented in custom MATLAB code for this paper and discussed in some detail in the previous section), the recognizer performance was also evaluated using wavelet feature vectors. This feature extraction method uses the wavelet transform, which represents the speech sample in terms of multiple timeshift and time-scalings of a selected finite-duration waveform or "wavelet." The theory of wavelet feature extraction will not be discussed in detail here. The bibliography includes an excellent text on this topic [22].

The wavelet feature vectors were calculated using the Periodized Biorthogonal Wavelet Transform function from the Princeton WAVELAB_802 library. This choice was based on published results [23] which demonstrated reliable speaker recognition using the biorthogonal discrete wavelet transform with a triangular shape decomposition scaling function. Those results found that analysis levels 3 through 8 of the wavelet transform provided reliable speaker identification information, while the first two levels tended to be somewhat noisy.

3. LMS RECOGNIZER

After feature extraction, the next phase of the speaker recognition process involves mapping each feature vector to a speaker. The mapping technique used in our baseline system uses an artificial neural network to map a cepstral vector representation of a speech sample to the member of the speaker set most likely to have produced the speech sample. In order to train the neural network and evaluate its performance, the feature extraction algorithm described above is used to produce two sets of feature vectors for each member of the speaker set. One set of feature vectors is used to train the neural network and the second set is used as test data to evaluate the recognition performance of the mapping network.

In practice, the test data might be collected years after the training data and might be recorded in a significantly different environment. For the baseline (LMS) recognizer demonstration discussed in this section, the two data sets were collected using the same recording equipment, in the same environment, and with minimal background noise. The more sophisticated recognizer implementations discussed in Sections 6-8 used more realistic recordings from a widely-used speech corpus to evaluate the impact of recording quality and background noise.

There is a wide variety of neural network architectures and training algorithms that can be used to map an N-element feature vector to one member of an M-element speaker set. For the baseline recognizer, we have chosen a relatively simple single-layer network using a Least Mean Square (LMS) training algorithm [24].

A separate neural network is trained for each member of the speaker set. The networks are arranged in a parallel structure so that each network is presented with the same 15-element cepstral vector at each training or test interval. During the training phase, each network is trained to produce a large output when the input is from the speaker associated with that network and to produce a small output when the input in from any other speaker. During the test phase, the input vector is mapped to a speaker by choosing the speaker associated with the network that has the largest response to the test vector.

For the baseline recognizer, the training of each branch of the parallel network is accomplished separately. The training process begins by initializing each synaptic weight to a small random number. The input to the network is then presented with feature vectors which alternate between vectors from the “desired” speaker and vectors from one of the other speakers. The LMS training method is illustrated in Figure 3-1.

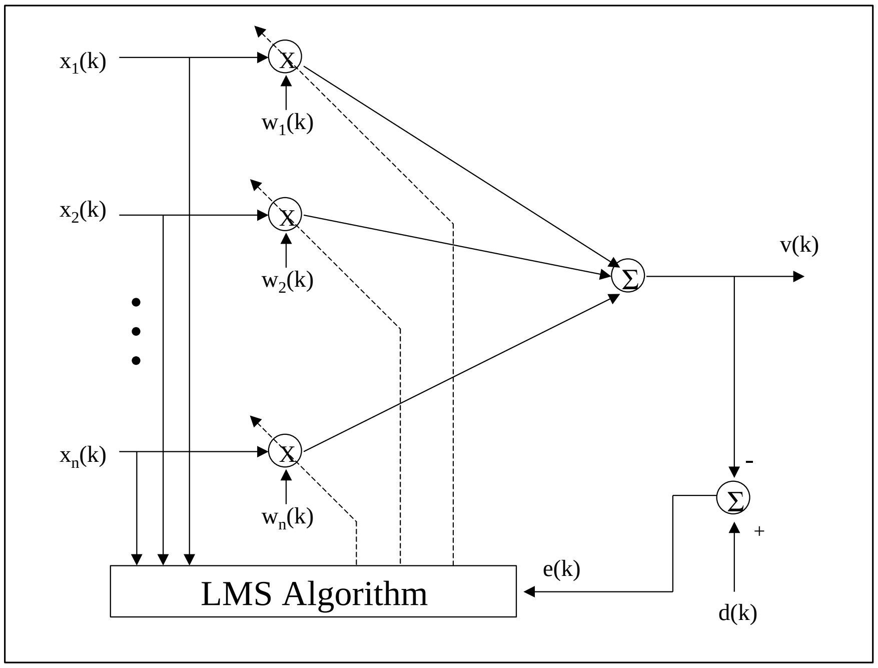

Figure 2-1 LMS Training Algorithm

At each training interval, the weights are adjusted as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(k)e(k)\mathbf{x}(k) \tag{13}$$

where: $\mathbf{w}(k)$ is the synaptic weight vector at training interval k,

$\mathbf{x}(k)$ is the input feature vector at training interval k,

e(k) is the "error" at interval k defined as $e(k) = d(k) - \mathbf{w}(k)\mathbf{x}(k)$

d(k) = 1 when the input feature vector is from the desired speaker

d(k) = 0 when the input feature vector is not from the desired speaker

and $\mu(k)$ is the "Learning Rate Parameter" defined as follows:

$$\mu(0) = 0.9/\lambda \tag{14}$$

where λ is the largest eigenvalue of the covariance matrix formed by the input vectors, and

$$\mu(k+1) = \mu(k)/(1 + k/\tau) \tag{15}$$

where τ determines the rate of adaptation of the learning rate parameter and is set to 200 for our demonstration.

After the network training phase is complete (evidenced by the stabilization of the synaptic weights), a separate set of feature vectors is used to test the performance of the network. During the test phase, an input from an "unknown" speaker is presented to the input of the parallel network and the most likely speaker is chosen as the member of the speaker set associated with the network branch which produces the largest output. The key performance measure for the system is the probability of correct speaker recognition.

4. DESCRIPTION OF THE NEGATIVE REINFORCEMENT TRAINING ALGORITHM

The background and approach discussions above are all based on previously published work in the field of speaker recognition. We now move on to the algorithm improvements that form the basis of the paper.

In the baseline system, each of the neural network stages in the parallel recognizer is trained independently by applying the LMS algorithm while alternately presenting a feature vector from the speaker associated with that stage, then a feature vector from another speaker in the speaker set. When the feature vector is from the speaker for which the network is being trained, the desired output is set to "1" and when it is from another speaker, the desired output is set to "0." After the weights stabilize for one stage, the same algorithm is used to train the next stage.

The proposed training algorithm trains all the stages in parallel. The same feature vector is simultaneously presented to all the parallel stages. The LMS algorithm is applied as the input cycles through feature vectors from each speaker in the set. The desired output for the stage corresponding to the source of the current input vector is set to "1" and the desired output for all the other stages are set to "0." Each time the algorithm cycles through the M speakers in the speaker set, each stage of the recognizer is presented with one "true" input and M-1 "false" inputs, one from each of the other speakers in the set.

The results presented in Section 5 show that this training algorithm provides a significant improvement over the baseline system in terms of correct recognition rate. One drawback to this approach is that the entire network must be re-trained when a new member is added to the speaker set. Adding a speaker to the baseline system would only require training one more stage for the new speaker.

5. LMS RECOGNIZER PERFORMANCE

This section will present the results of a preliminary evaluation of the performance of the baseline (LMS) system and of the performance improvements afforded by the enhanced algorithm. This evaluation was made using a 3-member speaker set. Two separate recordings of the same recitation were made for each speaker: one for training data and one for test data.

Two versions of the LMS mapping program were implemented and evaluated. Both programs attempt to map the cepstral feature vectors from the feature extraction program to the speaker most likely to be the source of the speech sample.

The first version is the baseline recognizer. This version does not implement the negative reinforcement training and is used as a performance baseline against which to evaluate the proposed algorithm enhancement. The recognizer reads two data files generated by two separate executions of the feature extraction program. One data file contains training data and the second contains test data. The first phase of the program implements a LMS training algorithm to train each of the 3 parallel stages, in turn, using training data from the corresponding speakers. The second phase uses the test data to evaluate the performance of the recognizer in terms of correct recognition rate.

The second version adds the negative reinforcement training algorithm described in Section 4 to the baseline recognizer.

The same training and test data sets were used to evaluate both recognizer versions. The measured performance of the recognizers is summarized in Tables 5-1 and 5-2. Sample convergence plots for the synaptic weights are presented in Figure 5-1. The columns of the performance tables represent the recognizer's estimate of the speaker while the rows represent the actual speaker. For example, in Table 5-1, when the actual speaker was Speaker 1, the recognizer correctly identified Speaker 1 a total of 179 times. The recognizer incorrectly identified Speaker 1 as Speaker 2 at total of 17 times. This format for reporting performance of a mapping network is sometimes referred to as a "confusion matrix."

As we can see from the Tables, the addition of the negative reinforcement algorithm to the baseline system improved the overall correct recognition rate from 85.5% to 94.0%.

Table 5-1. Confusion Matrix for Baseline Recognizer

Correct Recognition Rate = 85.5%		**Speaker Prediction**		
		1	**2**	**3**
Actual Speaker	**1**	179	17	0
	2	18	152	26
	3	1	23	172

Table 5-2 Confusion Matrix for Recognizer with Negative Reinforcement

Correct Recognition Rate = 94.0%		**Speaker Prediction**		
		1	**2**	**3**
Actual Speaker	**1**	196	0	0
	2	13	164	19
	3	0	3	193

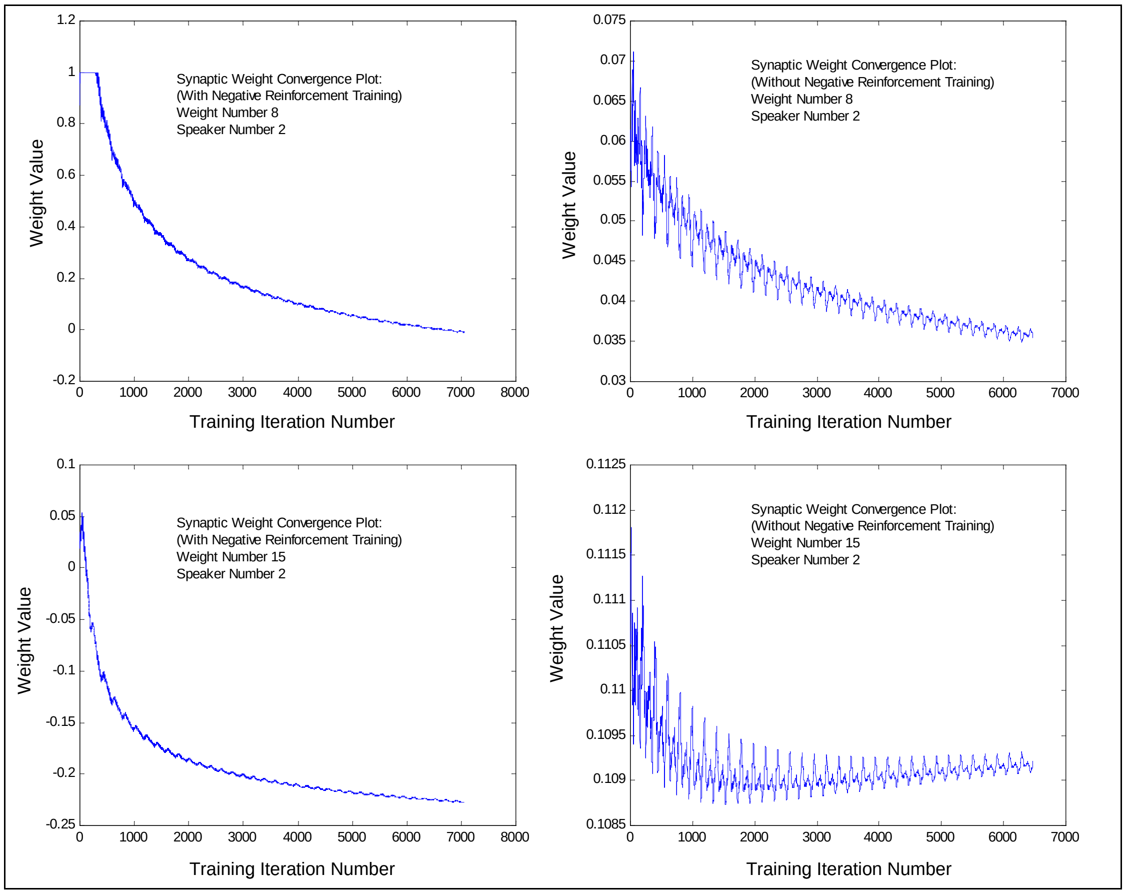

Figure 5-1 Synaptic Weight Convergence Plots

6. MULTI-LAYER PERCEPTRON RECOGNIZER

6.1 MLP Recognizer Description

The results presented in the previous section demonstrated the performance enhancement provided by the proposed recognizer improvements using a very simple single-layer recognizer implementing the LMS training algorithm. We now move on to more sophisticated ANN implementations beginning with the Multi-Layer Perceptron (MLP). The MLP architecture (Figure 6-1) uses several layers of neurons including an input layer, one or more "hidden" layers, and an output layer. The figure is representative of the recognizer implemented for this paper, which supports a variable length input vector, two hidden layers with variable number of neurons, and a scalar output. In general, the MLP recognizer may have any number of hidden layers and the output layer may be n-dimensional.

As with the LMS recognizer discussed earlier, the goal of the MLP recognizer is to map feature vectors to their source speakers. The parallel architecture is similar to that illustrated in Figure 1, consisting of a bank of recognizers each of which is trained to produce a large response for its associated speaker and a small response for all other speakers in the speaker set.

During the training phase, each parallel recognizer branch is presented with a succession of training vectors and the weight matrices are adjusted according to a backpropagation training algorithm which we will now present without derivation. The derivation of this and the other training algorithms used in the project are included in the referenced text by Ham and Kostanic [24].

Referring to Figure 6-1, let $x_i(k)$ represent the ith element of the input vector at training interval k, and let $w_{ij}^{(s)}(k)$ represent the synaptic weight corresponding to the synapse which connects the ith input element to the jth neuron for layer *s*. Then, the synaptic weights for layer *s* form a weight matrix $W^{(s)}$.

Up to this point, the MLP layer is similar to the LMS network presented earlier, with successive input vectors multiplying a weight matrix which is adapted to achieve a desired output. However, the MLP network introduces the complication of a non-linear "activation" function $f^{(s)}(\cdot)$ which seeks to better model the behavior of biological neurons. Our MLP recognizer uses the binary sigmoid function, a typical activation function for this application. This results in a small output from the artificial neuron until the activity level (input) to the neuron reaches a certain amplitude. The neuron then "fires" producing a significant output in response to a small change in the input.

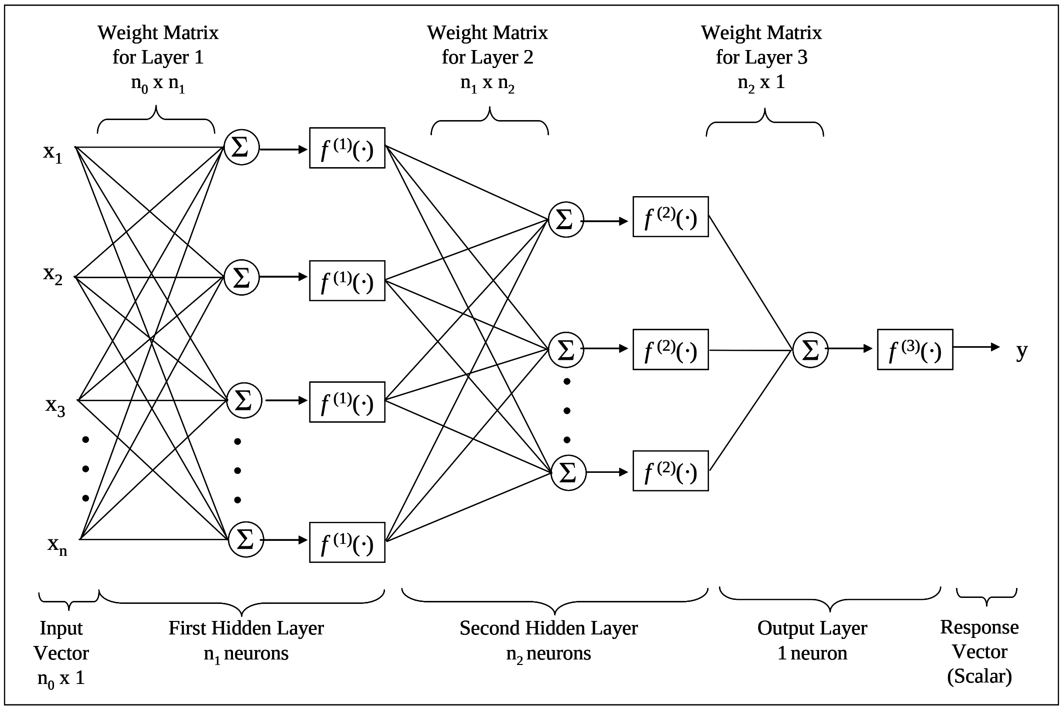
Weight Matrix for Layer 1 n_0 x n_1
Weight Matrix for Layer 2 n_1 x n_2
Weight Matrix for Layer 3 n_2 x 1
x_1
x_2
x_3
x_n
Σ
$f^{(1)}(\cdot)$
$f^{(2)}(\cdot)$
$f^{(3)}(\cdot)$
y
Input Vector n_0 x 1
First Hidden Layer n_1 neurons
Second Hidden Layer n_2 neurons
Output Layer 1 neuron
Response Vector (Scalar)

Figure 6-1 Multi-Layer Perceptron Block Diagram

Let $v^{(s)} = W^{(s)}x^{(s)}$ be the vector of activation levels at layer s. Then the output of the neuron will be $f^{(s)}(v_i)$, which, for the binary sigmoid activation function is:

$$f_{bs}(v) = \frac{1}{1+e^{-av}} \tag{16}$$

Where *a* is a constant which determines the abruptness of the activation.

The training algorithm also involves the first derivative of the activation function, $g^{(s)}(v)$, which, for the binary sigmoid activation function is:

$$g_{bs}(v) = a f_{bs}(v)[1 - f_{bs}(v)] \tag{17}$$

The training algorithm begins by setting all the synaptic weights to small random numbers. The feature vectors calculated by the feature extraction algorithm are then presented to the network as inputs, cycling through vectors form each of the speakers in the speaker set. The goal of the algorithm is to adjust the weights so that the final output of the network is large when the input corresponds to a particular speaker and small when the input is from any other speaker. Accordingly, the algorithm is deigned to minimize the “error” in each layer - That is, to minimize the difference between the actual output of each layer and the desired output.

The learning rule for the MLP network is [24]:

$$w_{ji}^{(s)}(k+1) = w_{ji}^{(s)}(k) + \mu^{(s)}\delta_j^{(s)} x_{out,i}^{(s)} \qquad (18)$$

where

$$\delta_j^{(s)} = (d - x_{out,j}^{(s)})g(v_j^{(s)}) \qquad (19)$$

for the final output layer and

$$\delta_j^{(s)} = (\sum_{h=1}^{n_s+1}\delta_h^{(s+1)} w_{hj}^{(s+1)})g(v_j^{(s)}) \qquad (20)$$

for the hidden layers.

6.2 TIMIT Speech Corpus

The LMS recognizer performance discussed in the previous sections was accomplished using the prototype recognizer and used ".wav" formatted recordings for training and test data. These recordings were adequate to demonstrate the concept of the proposed recognizer improvements. However, the recordings were practically noise-free and represented a very small speaker set. To train and test the MLP and other advanced recognizers, we need a more realistic set of speech data.

The Texas Instruments / Massachusetts Institute of Technology (TIMIT) Acoustic-Phonetic Continuous Speech Corpus, was commissioned by the National Institute of Standards and Technology (NIST) for just this purpose.

The subset of the TIMIT corpus used for this evaluation consists of speech recordings of 12 speakers recorded in a realistic background noise typical of an office environment. For a full description of the TIMIT speech corpus and information on ordering the CD-ROM, see the NIST report [25].

6.3 Explanation of Receiver Operating Curves

The performance results from the prototype LMS recognizer presented in Section 5 were based on simply selecting the predicted speaker based on which parallel module had the largest output. In practice, the decision threshold for each parallel module may be different. The selection of these thresholds is based on the relative cost of having a false detection (false alarm) vs. the cost of a missed detection. If the threshold is set very low, we never miss a detection but we have a high false alarm rate. If the threshold is set too high, we don't have any false alarms but we have a high rate of missed detections. The Receiver Operating Curve (ROC) is a useful tool for presenting performance of a recognizer in these terms. The ROC is simply a plot of Probability of False Alarm vs.

Probability of Correct Detection as threshold is varied from 0 to infinity. Obviously, the ideal recognizer is one which achieves 100% correct detection with 0% false alarms. So, the figure of merit for our recognizers is the degree to which they approach this ideal performance.

6.4 MLP Recognizer Performance

The MLP recognizer was implemented and tested using custom MATLAB code which supports a variable number of neurons in each of the two hidden layers. The test results presented here are based on 15 neurons in the first hidden layer and 5 in the second hidden layer, each with a binary sigmoid activation function.

The performance for the MLP recognizer (and the remaining recognizers) will be presented in terms of ROC curves representing recognizer performance against a selected set of speakers. To facilitate comparison of performance, the same training and test data was used for the MLP recognizer and for the other two advanced recognizers to be presented in the following sections. The training and test data consists of 80 speech samples (40 for training data and 40 for test data) from each of 12 speakers selected from the TIMIT speech corpus. This provides a total of 480 test samples for each ROC plot. Performance will be presented and compared for each of the three recognizer algorithms (MLP, RBF, and PLSR), and for both of the feature extraction algorithms (cepstral and wavelet feature vectors). The ROC plots for the MLP recognizer are presented in Figures 6-2 and 6-3.

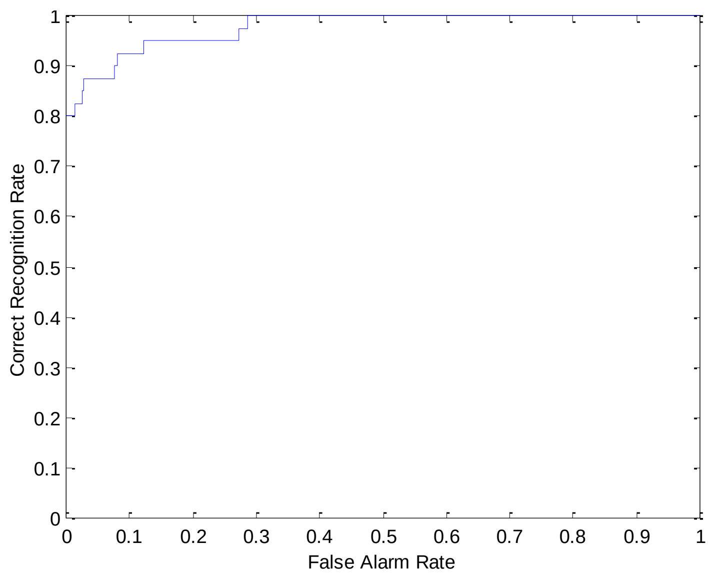

Figure 6-2. ROC Plot, MLP Binary Sigmoid, Speaker 2, Cepstral Features

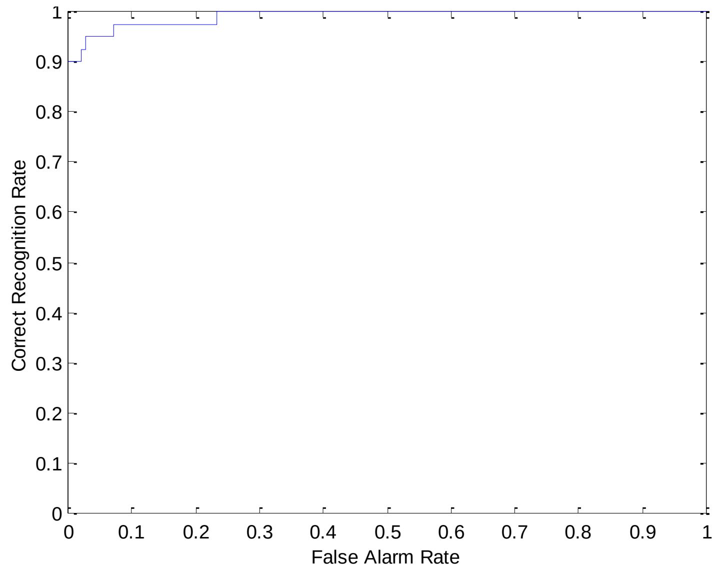

Figure 6-3. ROC Plot, MLP Binary Sigmoid, Speaker 6, Cepstral Features

7. RADIAL BASIS FUNCTION RECOGNIZER

7.1 RBF Recognizer Description

The next mapping network we will consider is the Radial Basis Function (RBF) recognizer. The RBF recognizer block diagram is shown in Figure 7-1. The RBF network consists of an input layer, a single hidden layer, and an output layer. As always, the goal is to iteratively adjust the network parameters during the training phase so that the network will produce a large output when the input vector is from its associated speaker and a small output otherwise.

The network design involves selecting an RBF activation function, ϕ_k, a set of initial "center vectors", c_k , and corresponding "spread parameters", σ_k. While a number of options are available for the functional form of $\phi_k(\cdot)$ [24], the form chosen for our implementation is the Gaussian function:

$$\phi(x, c_k) = \exp(-\frac{\|x - c_k\|^2}{\sigma_k^2}) \tag{21}$$

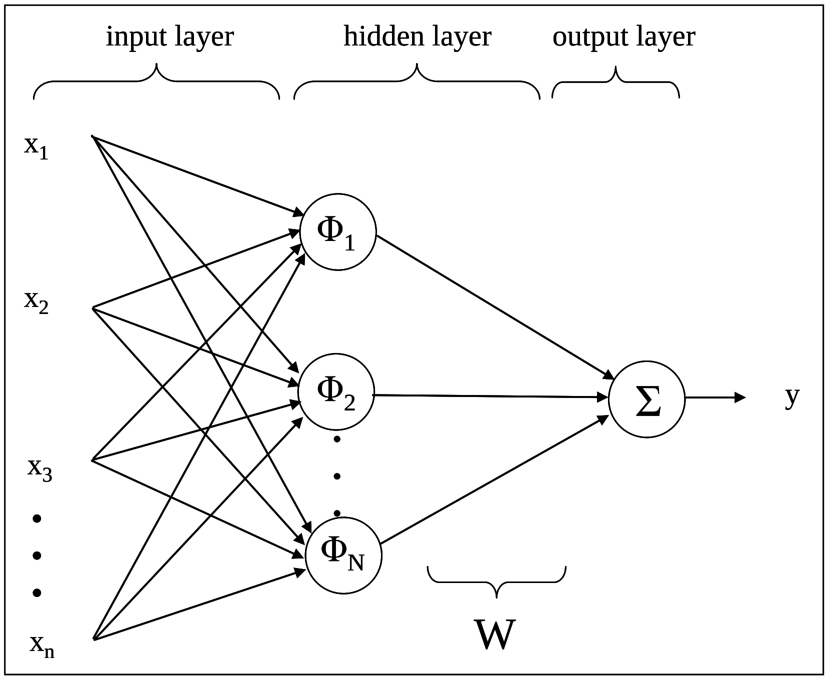

Figure 7-1 RBF Recognizer Block Diagram

The center vectors, one for each neuron in the hidden layer, are intended to provide a representative sampling of the input vector space. The most common approach to selecting the initial center vectors, and the approach used in our RBF implementation, is to select the centers at random from the set of training vectors. Once the center vectors have been chosen, the initial values for the spread parameters for each center are calculated according to:

$$\sigma = \frac{d_{\max}}{\sqrt{K}} \tag{22}$$

where d $_{\max}$ is the maximum Euclidian distance between the selected center and all the other centers and K is the number of centers.

The training algorithm begins by initializing the weights in the output layer to small random values, then iteratively calculates the network parameters according to the Stochastic Gradient method:

Present an input (feature vector) and calculate the network output:

$$\hat{y}(n) = \sum_{k=1}^{N} w_k \phi\{x(n), c_k, \sigma_k\} \tag{23}$$

Update network parameters according to:

$$w(n+1) = w(n) + \mu_w e(n)\psi(n) \tag{24}$$

$$c_k(n+1) = c_k(n) + \mu_c \frac{e(n)w_k(n)}{\sigma_k^2(n)} \phi\{x(n), c_k(n), \sigma_k\}[x(n) - c_k(n)] \tag{25}$$

$$\sigma_k(n+1) = \sigma_k(n) + \mu_\sigma \frac{e(n)w_k(n)}{\sigma_k^3(n)} \phi\{x(n), c_k(n), \sigma_k\}[x(n) - c_k(n)] \tag{26}$$

where

$$\psi(n) = [\phi\{x(n), c_1, \sigma_1\}, \quad \phi\{x(n), c_2, \sigma_2\}, \quad \cdots \quad \phi\{x(n), c_N, \sigma_N\}]^T \tag{27}$$

$$e(n) = \hat{y}(n) - y_d(n) \tag{28}$$

7.2 RBF Recognizer Performance

The RBF recognizer was implemented and tested using custom MATLAB code which supports a variable number of neurons in the hidden layer. The test results presented here are based on 15 neurons in the hidden layer, each with a Gaussian Radial Basis function.

The RBF recognizer was trained an tested using speech data from 12 speakers selected from the TIMIT speech corpus (the same data that was used for the MLP recognizer). The ROC plots for the RBF recognizer are presented in Figures 7-2 and 7-3.

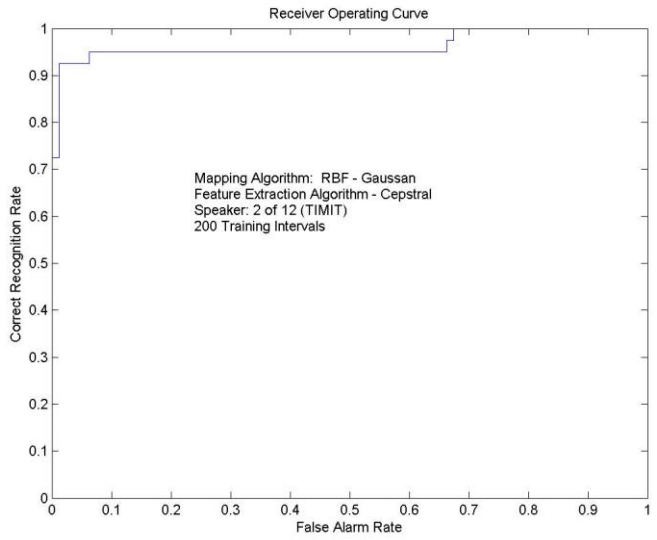

Figure 7-2. ROC Plot: RBF Gaussian, Speaker 2, Cepstral Features

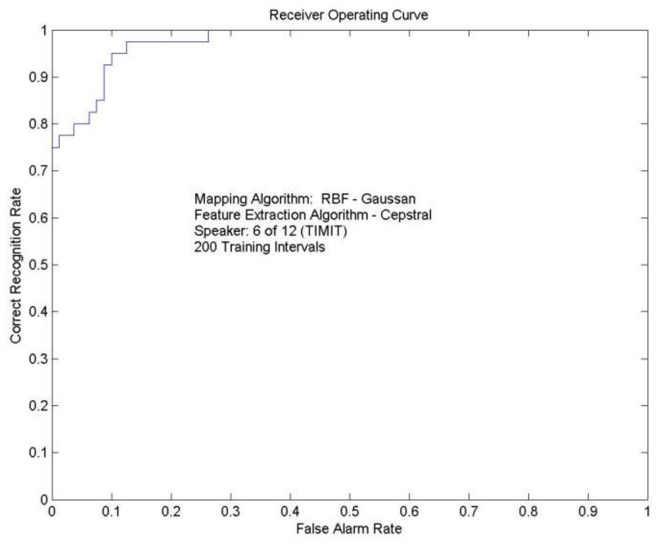

Figure 7-3. ROC Plot: RBF Gaussian, Speaker 6, Cepstral Features

8. PLSR RECOGNIZER

8.1 PLSR Recognizer Description

The Partial Least Squares Regression (PLSR) algorithm is one of a family of analysis methods known as "factor analysis" methods. The goal of the PLSR algorithm is to find a "calibration model" which predicts certain physical quantities related to a set of empirical measurements. In our case, the empirical measurements are the feature vectors extracted from training samples of speech data and the quantity to be predicted is the likelihood that the feature vector corresponds to a particular speaker.

We begin by defining a data matrix **A** containing all the empirical measurements from the training data. Each row of **A** contains a feature vector from one of the speakers, alternating through the speakers in the speaker set. So if the total number of measurements (the number of speakers times the number of training samples per speaker) is m and the number of elements in the feature vector is n, we can write $\boldsymbol{A} \in \Re^{m \times n}$. The same **A** matrix is used for all the parallel modules.

Next, we define a "target vector" $c \in \Re^{m \times 1}$ so that each element of **c** corresponds to a row of **A**.. Each parallel module has a different target vector populated so that $c_m = 1$, if the corresponding measurement in A is from the speaker associated with the module and $c_m = 0$ otherwise. Our goal is to find a calibration model $\hat{b}_{f-PLSR} \in \Re^{n \times 1}$, which produces a reliable estimate of **c** when employed during the test phase:

$$\hat{c}_{test} = A_{test}\,\hat{b}_{f-PLSR} \tag{29}$$

8.2 PLSR1 Calibration Algorithm

An iterative approach, referred to as the PLSR1 calibration algorithm [24], is used to produce the calibration model for each of the parallel modules. The following steps comprise the PSLR1 algorithm:

Step 1: Mean-Center and Variance-Scale the training data. For each column of **A**, calculate the mean and standard deviation of the elements in the column. Subtract the mean value of the column from each element in the column. Then, divide each element by the standard deviation of the column. Set parameter h to 1 (where h represents the number of PLS factors).

Step 2: Form the weight-loading vector:

$$\hat{\boldsymbol{w}}_h = \frac{\boldsymbol{A}^T \mathbf{c}}{\mathbf{c}^T \mathbf{c}} \tag{30}$$

then, normalize the weight-loading vecor:

$$\hat{\mathrm{w}}_h \leftarrow \frac{\hat{\mathrm{w}}_h}{\left\|\hat{\mathrm{w}}_h\right\|_2} \tag{31}$$

Step 3: Generate the “score” vector:

$$\hat{\boldsymbol{t}}_h = \boldsymbol{A}\,\hat{\boldsymbol{w}}_h \tag{32}$$

Step 4: Calculate the Regression Coefficient:

$$\hat{v}_h = \hat{\boldsymbol{t}}_h \frac{\hat{\boldsymbol{t}}_h^T \mathbf{c}}{\hat{\boldsymbol{t}}_h^T \hat{\boldsymbol{t}}_h} \tag{33}$$

Step 5: Calculate the Loading Vector:

$$\hat{\boldsymbol{b}}_h = \frac{\boldsymbol{A}^T \hat{\boldsymbol{t}}_h}{\hat{\boldsymbol{t}}_h^T \hat{\boldsymbol{t}}_h} \tag{34}$$

Step 6: Calculate the residuals in ***A*** and ***c***.

$$\boldsymbol{E}_A = \boldsymbol{A} - \hat{\boldsymbol{t}}_h \hat{\boldsymbol{b}}_h^T \tag{35}$$

$$\boldsymbol{e}_c = \boldsymbol{c} - \hat{v}_h \hat{\boldsymbol{t}}_h \tag{36}$$

Step 7: Increment h, repeat from Step 2, substituting E_A for A and e_c for c. Continue until h reaches the optimal value – producing the minimum Standard Error of Prediction. This value is determined empirically be calculating SEP for each pass through the algorithm.

8.3 PLSR Recognizer Performance

The RBF recognizer was implemented and tested using custom MATLAB code which is based on a PLSR1 calibration algorithm using 10 PLS factors.

The PLSR recognizer was trained and tested using data from 12 speakers selected from the TIMIT speech corpus. The ROC plots for the PLSR recognizer are presented in Figures 8-1 and 8-2.

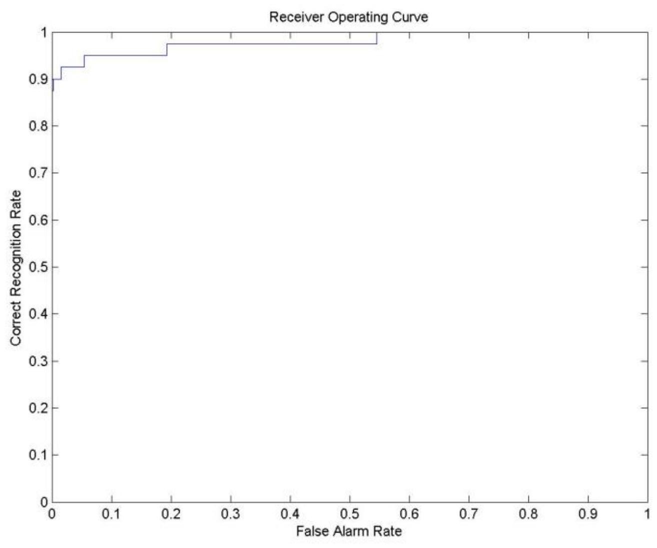

Figure 8-1. ROC Plot: PLSR, Speaker 2, Cepstral Features

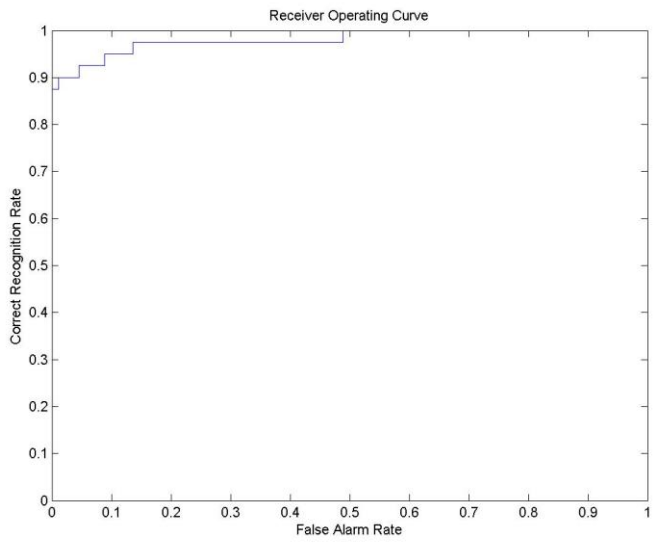

Figure 8-2. ROC Plot: PLSR, Speaker 6, Cepstral Features

9. CONCLUSIONS

This paper presented a novel negative reinforcement training algorithm for neural network speaker recognition systems and demonstrated that the performance of this algorithm is significantly improved over a similarly configured recognizer that does not employ negative reinforcement training. The performance of the algorithm was presented in terms of Receiver Operating Curves for two separate feature extraction schemes (cepstral vector and wavelet transform) and four separate classifier algorithms (LMS, MLP, RBF, and PLSR).

The prototype (LMS) implementation of the proposed recognizer enhancement significantly improved the performance over the baseline system in terms of probability of correct recognition. The addition of the negative reinforcement algorithm to the baseline LMS system improved the overall correct recognition rate from 85.5% to 94.0%.

All three of the advanced mapping algorithms – MLP, RBF, and PLSR – consistently provided over 90% correct recognition rate with less than 10% false alarm rate when operating on cepstral feature vectors. Table 9-1 summarizes the performance of the three advanced mapping algorithms using each of the feature extraction methods in terms of Probability of Correct Detection (P_{cd}) vs. Probability of False Alarm (P_{fa}). It is important to note that these numbers are based on the probability of mapping a single cepstral feature vector to the correct member of the speaker set using a standard NIST speaker corpus and realistic recording conditions. Since a brief speech recording could easily produce thousands of samples of the feature vector, any of the three advanced recognizer algorithms could achieve near perfect recognition performance with this speaker set and recording conditions.

All three of the mapping algorithms demonstrated significantly better performance when operating on cepstral feature vectors vs. wavelet feature vectors. This result is consistent with results from other recent research [1, 3, 6, 16, 17], and underscores the power of the cepstral vector algorithm as a speaker recognition tool.

Table 9-1 Recognizer Performance Summary

	Feature Vector Option			
	Cepstral		Wavelet	
	P_{cd}	P_{fa}	P_{cd}	P_{fa}
Multi-Layer Perceptron	94%	8%	85%	10%
Radial Basis Function	93%	6%	74%	18%
Partial Least Squares Regression	95%	5%	82%	16%

It is somewhat difficult to compare performance with existing speaker recognition systems due to the variability in training and test conditions used in the published work.

Variables affecting the performance include number of speakers in the speaker set, length of recording samples used, recording environment, etc. We have attempted to quantify these variables in our results by using a standard (TIMIT) speech corpus and presenting the results in terms of ROC curves based on single feature vector tests. The best performance benchmark we found is an overview paper published by MIT-LL [24] which shows typical performance for recognizers operating on “conversational speech” with 2 minutes of training data and 30 seconds of test data to be in the range of 7% - 15% probability of missed detection with an equal false alarm rate. Admittedly, this information is slightly dated (publication date is 2002), but we have found no documentation of major breakthroughs in performance since that date. Since all three of our recognizers (MLP, RBF, and PLSR) matched the upper end of this range while operating on single cepstral feature vectors (representing a fraction of a second of test data), we believe there is good evidence of substantial performance improvement using the proposed negative reinforcement algorithm.

There are several areas related to this topic that could benefit from additional research. First, it would be interesting to determine the degree to which performance could be further improved through the use of more complex recognizer architectures (e.g., increasing the number of hidden layers and/or the number of neurons in each layer for the MLP recognizer). Secondly, our RBF recognizer was implemented using the Gaussian basis function, but there are several others that could be implemented for comparison (e.g., cubic approximation function, thin-plate spline function, or multiquadratic function). Finally, we have stated that our results were obtained using test samples consisting of a single feature vector. Since a brief sample of speech could easily produce hundreds of feature vectors, further research should be conducted to determine the performance improvement achievable by implementing a decision scheme based on multiple feature vectors. For example, a majority vote decision scheme might be implemented to minimize spurious false alarms.

REFERENCES

[1] H. Seddik, A. Rahmouni, M. Sayadi. “Text independent speaker recognition using the mel frequency cepstral coefficients and a neural network classifier” *First International Symposium on Control, Communications and Signal Processing*, March 21-24, 2004, pages 631 – 634.

[2] X. Yue, D. Ye, C. Zheng, and X. Wu. “Neural networks for improved text-independent speaker identification” *Engineering in Medicine and Biology Magazine, IEEE* , Volume: 21 , Issue: 2 , Mar/Apr 2002, pages 53 – 58.

[3] S. Guruprasad, N. Dhananjaya, and B. Yegnanarayana. “AANN Models For Speaker Recognition Based On Difference Cepstrals” *Proceedings of the International Joint Conference on Neural Networks*, Volume: 1 , 20-24 July 2003 pages 692 - 697.

[4] F. Mueen, A. Ahmed, A. Gaba. “Speaker recognition using artificial neural networks” *Students Conference, ISCON '02, Proceedings IEEE* , 16-17 Aug. 2002, Volume 1, pages:99 – 102.

[5] V. Moonasar, G.K. Venayagamoorthy. “A committee of neural networks for automatic speaker recognition (ASR) systems” *International Joint Conference on Neural Networks, IJCNN '01*, 15-19 July 2001, Volume 4, pages 2936 – 2940.

[6] Kajarekar, S.S. "Four Weightings and a Fusion: A Cepstral-SVM System for Speaker Recognition" *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on* , vol., no.pp. 225- 230, Nov. 27, 2005.

[7] Porwal, G.; Patil, H.A.; Basu, T.K. "Effect of speech coding on text-independent speaker identification" *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on* , vol., no.pp. 415- 420, 4-7 Jan. 2005

[8] Zilca, R.D.; Kingsbury, B.; Navratil, J.; Ramaswamy, G.N. “Pseudo Pitch Synchronous Analysis of Speech With Applications to Speaker Recognition” *Audio, Speech and Language Processing, IEEE Transactions on,* Volume 14, Issue 2, March 2006 Page(s):467 – 478.

[9] Eriksson, T.; Kim, S.; Hong-Goo Kang; Chungyong Lee. “An information-theoretic perspective on feature selection in speaker recognition” *Signal Processing Letters, IEEE,* Volume 12, Issue 7, July 2005 Page(s):500 – 503.

[10] Faundez-Zanuy, M.; Monte-Moreno, E. “State-of-the-art in speaker recognition” *Aerospace and Electronic Systems Magazine, IEEE,* Volume 20, Issue 5, March 2005 Page(s):7 – 12.

[11] Kodukula, S.R.M.; Mahadeva Prasanna, S.R.; Yegnanarayana, B. “Neural network models for extracting complementary speaker-specific information from residual phase” *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing.,* 4-7 Jan. 2005 Page(s):421 – 425.

[12] Jing Deng; Thomas Fang Zheng; Zhan-Jiang Song; Jian Liu; Wen-Hu Wu. “Using predictive differential power spectrum and subband mel-spectrum centroid for robust speaker recognition in stationary noises” *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on,* Volume 8, 18-21 Aug. 2005 Page(s):4846 – 4851.

[13] Yong-Qiang Bao; Li Zhao; Cai-Rong Zou. “Study on speaker recognition under noise environments based on PCANN” *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on,* Volume 6, 26-29 Aug. 2004 Page(s):3770 – 3774.

[14] Wanfeng Zhang; Yingchun Yang; Zhaohui Wu. “Exploiting PCA classifiers to speaker recognition” *Neural Networks, 2003. Proceedings of the International Joint Conference on,* Volume 1, 20-24 July 2003 Page(s):820 – 823.

[15] Reynolds, D.A. "An overview of automatic speaker recognition technology" *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on,* Volume 4, 13-17 May 2002 Page(s):IV-4072 - IV-4075.

[16] Siew Chan Woo; Chee Peng Lim; Osman, R. "Development of a speaker recognition system using wavelets and artificial neural networks" *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, 2-4 May 2001 Page(s):413 – 416.

[17] George, S.; Dibazar, A.; Liaw, J.-S.; Berger, T.W. "Speaker recognition using dynamic synapse based neural networks with wavelet preprocessing" *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on,* Volume 2, 15-19 July 2001 Page(s):1122 – 1125.

[18] Inal, M.; Fatihoglu, Y.S. "Self organizing map and associative memory model hybrid classifier for speaker recognition" *Neural Network Applications in Electrical Engineering, 2002. NEUREL '02. 2002 6th Seminar on*, 26-28 Sept. 2002 Page(s):71 – 74.

[19] Hsieh, C.-T.; Lai, E.; Wang, Y.-C. "Robust speech features based on wavelet transform with application to speaker identification" *Vision, Image and Signal Processing, IEE Proceedings,* Volume 149, Issue 2, April 2002 Page(s):108 – 114.

[20] H. Avi-Itzhak and T. Diep. "Arbitrarily tight upper and lower bounds on the Bayesian probability of error" *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*-18(1):89-91, 1996.

[21] F.M. Ham and S. Park. "A Robust Neural Network Classifier for Infrasound Events Using Multiple Array Data" *Proceedings of the 2002 World Congress on Computational Intelligence – International Joint Conference on Neural Networks*, Honolulu, Hawaii, May 12-17, 2002, pp. 2615-2619.

[22] G. Strang and T. Nguyen, *Wavelets and Filter Banks,* Wellesley-Cambridge Press, Wellesley MA, 1996.

[23] F. Farahani, P.G. Georgiou, and S.S. Narayanan. "Speaker Identification Using Supra-Segmental Pitch Pattern Dynamics" *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, 2004.

[24] F.M. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering*, McGraw-Hill, New York, 2001.

[25] J.S. Garofolo, et.al. "TIMIT Acoustic-Phonetic Continuous Speech Corpus" *National Institute of Standards and Technology, NISTIR* 4930, 1993.