Speaker Recognition Using Neural Networks and Conventional Classifiers

Kevin R. Farrell, *Member, IEEE,* Richard J. Mammone, *Senior Member, IEEE,* and Khaled T. Assaleh, *Member, IEEE*

***Abstract*— An evaluation of various classifiers for text-independent speaker recognition is presented. In addition, a new classifier is examined for this application. The new classifier is called the modified neural tree network (MNTN). The MNTN is a hierarchical classifier that combines the properties of decision trees and feedforward neural networks. The MNTN differs from the standard NTN in both the new learning rule used and the pruning criteria. The MNTN is evaluated for several speaker recognition experiments. These include closed- and open-set speaker identification and speaker verification. The database used is a subset of the TIMIT database consisting of 38 speakers from the same dialect region. The MNTN is compared with nearest neighbor classifiers, full-search, and tree-structured vector quantization (VQ) classifiers, multilayer perceptrons (MLP's), and decision trees. For closed-set speaker identification experiments, the full-search VQ classifier and MNTN demonstrate comparable performance. Both methods perform significantly better than the other classifiers for this task. The MNTN and full-search VQ classifiers are also compared for several speaker verification and open-set speaker-identification experiments. The MNTN is found to perform better than full-search VQ classifiers for both of these applications. In addition to matching or exceeding the performance of the VQ classifier for these applications, the MNTN also provides a logarithmic saving for retrieval.**

I. Introduction

THE objective of speaker identification is to determine which speaker is present based on the individual's utterance. This is in contrast with speaker verification, where the objective is to verify the person's claimed identity based on his or her utterance. Speaker identification and speaker verification fall under the general category of speaker recognition [1]–[4]. A generic speaker recognition system is shown in Fig. 1. In Fig. 1, the desired features are first extracted from the speech signal. The extracted features are then used as input to a classifier, which makes the final decision regarding verification or identification.

Speaker identification systems can be closed set or open set. Closed-set speaker identification refers to the case where the speaker is known *a priori* to be a member of a set of N speakers. Open-set speaker identification includes the additional possibility where the speaker may be from outside the set of N speakers. Open-set speaker identification and speaker verification often use thresholding to determine if a speaker is out of the set.

Manuscript March 4, 1993; revised September 15, 1993. This work was funded in part by the Air Force/Rome Laboratories.

The authors are with the CAIP Center, Rutgers University, Piscataway, NJ 08855.

IEE Log Number 9214711.

Fig. 1. Speaker recognition system.

Another distinguishing feature of speaker recognition systems is whether they are text dependent or text independent. Text-dependent speaker recognition systems require that the speaker utter a specific phrase or a given password. Text-independent speaker identification systems identify the speaker regardless of his utterance. This paper focuses on the text-independent speaker identification and speaker verification tasks.

A new classifier is introduced and evaluated for speaker recognition. The new classifier is the modified neural tree network (MNTN). The MNTN incorporates modifications to the learning rule of the original NTN [5] and uses a new pruning criteria. In addition, nearest neighbor (NN) classifiers, vector quantization (VQ) classifiers, multilayer perceptrons (MLP's), and decision trees are evaluated for comparison.

This paper is organized as follows. Section II reviews feature extraction for speaker recognition. Section III presents the traditional classifiers considered here for text-independent speaker recognition. Section IV reviews the NTN and discusses the modifications. Section V describes the database used and the experiments. The summary and conclusions of the paper are given in Section VI.

II. Feature Extraction

The process of feature extraction consists of obtaining characteristic parameters of a signal to be used to classify the signal. The extraction of salient features is a key step in solving any pattern recognition problem. For speaker recognition, the features extracted from a speech signal should be invariant with regard to the desired speaker while exhibiting a large deviation from the features of an imposter.

The selection of speaker-unique features from a speech signal is an ongoing issue. It has been found that certain features yield better performance for some applications than do other features. Thus far, no feature set has been found to allow perfect discrimination for all conditions.

Early text-dependent speaker recognition systems utilized information from the short-time spectrum to provide speaker-unique features [6], [7]. These features consisted of energy measurements from the outputs of a bank of filters. Additional features that have been considered for speaker recognition

include pitch [8], formant frequencies [9], intensity [10], log-area ratios [11], and linear prediction (LP) coefficients [12]. Previous evaluation of feature sets [13], [14] have ranked various features and found the best speaker characteristics to be obtained from the pitch and specific formant locations in certain vowels and nasals. However, isolating these events during text-independent applications has proved to be difficult. Atal [15] provided a comparison of parameters obtained from linear prediction, the impulse response, autocorrelation, vocal tract area function, pitch, and cepstral coefficients and found the cepstrum to provide the best results for speaker recognition. Today, cepstral coefficients are the dominant features used for speaker recognition [16]–[18]. We shall use cepstral coefficients as a basis for comparing the different classifiers.

A. Cepstral Coefficients

The cepstral coefficients are given by the inverse Fourier transform of the logarithm of the power spectrum of a signal. The cepstral coefficients represent a decoupling of the glottal source and the vocal tract response for a speech signal [19]. Cepstral coefficients derived from the LP coefficients are generally used in speech and speaker recognition applications [16]–[18].

The cepstrum can be obtained directly from the frequency spectrum or from the LP coefficients. Both methods provide similar results for speaker recognition [16]. However, computing the LP-based cepstrum requires less computation and tends to be the method of choice.

The principle of LP analysis is that a sample of a time series, i.e., speech, can be approximated as a linear combination of past samples. This can be expressed as follows:

$$\hat{s}_n = \sum_{k=1}^{N} a_k s_{n-k} \tag{1}$$

where s_n are the speech samples, n is the time index, and a_k are the LP coefficients.

Given the LP coefficients a_k, $1 \leq k \leq N$, the cepstral coefficients c_k are usually obtained from the LP coefficients by the recursive relationship [15]:

$$\begin{aligned} c_1 &= a_1 \\ c_n &= \sum_{k=1}^{n-1} (1 - \frac{k}{n}) a_k c_{n-k} + a_n \\ 1 &< n < P. \end{aligned} \tag{2}$$

If the dimension of the cepstral vector P is greater than N, then a_n is set to zero for $P < n \leq N$. The P-dimensional vector is used as the input to the classifier. Variants of the cepstral coefficients, such as the delta cepstrum [20] and adaptive component weighted (ACW) cepstrum [21], have also been found to perform well for noisy speech. We shall use the standard LP-derived cepstrum represented in (2).

III. Classification

Various classifiers have been considered for speaker recognition. Early methods consisted of using the Euclidean distance between the features of a test and reference utterance [6].

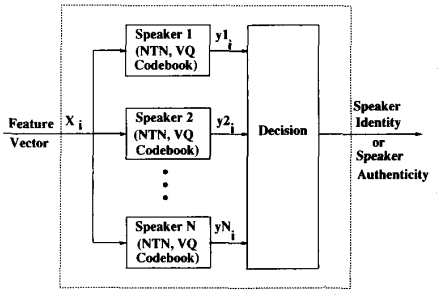


Fig. 2. Classifier structure for speaker recognition.

Later, it was shown that Mahalanobis and weighted distances could further improve classification [7], [22]. The VQ classifier was then applied to speaker recognition [17], [23] and has since been the benchmark classifier for text-independent speaker recognition systems. More recently, classical statistical pattern recognition approaches, such as Bayesian discriminant analysis [24] and nearest neighbor classifiers [25], have been considered for speaker recognition. Other alternative classifiers have included neural networks, such as multilayer perceptrons (MLP's) [26], time delay neural networks (TDNN's) [27], radial basis function (RBF) networks [28], learning vector quantization (LVQ) [29], and other forms of neural networks [30].

Currently, the most prominent speaker recognition classifiers, such as VQ, are based on unsupervised training algorithms. A VQ-based classifier that has become popular for speaker recognition is the discrete hidden Markov model (HMM) [32], [34]. Discrete HMM's augment temporal information to VQ to utilize the transition probabilities between states. Another unsupervised classifier is the Gaussian mixture model (GMM). This technique models each speaker as a sum of Gaussian mixtures [31]. The GMM can also use temporal information, i.e., the state transition probabilities, in which case a continuous density HMM [33] results. It has been claimed that the improvements due to the addition of temporal information are negligible for text-independent speaker recognition applications [34]. Classifiers that take advantage of the temporal correlations of the feature vectors, such as TDNN's and HMM's, are not considered here. Although temporal information can be of help in text-dependent applications, it will be reported elsewhere.

The classification stage of text-independent speaker recognition is typically implemented by modeling each speaker with an individual classifier. The classifier structure for speaker recognition is illustrated in Fig. 2.

In Fig. 2, each speaker model operates as a speaker verification system on a feature vector-by-vector basis. Given a specific feature vector, each speaker model associates a number corresponding to the degree of match with that speaker. This structure is applied to speaker recognition as follows. For closed-set speaker identification, all feature vectors of the test utterance are applied to each of the speaker models. The speaker is selected as having the model that is most likely to have generated those feature vectors. For speaker verification, all feature vectors for the test utterance are applied only to the speaker model for the speaker to

be verified. A measure is then made for the likelihood that these feature vectors were generated by that speaker model. If this measure exceeds a threshold, the speaker is verified or rejected. For open-set speaker identification, the feature vectors for the test utterance are applied to all speaker models, and the speaker is tentatively selected as the most likely model. The likelihood measure for this model is then compared with a threshold to determine if the corresponding speaker or "out of set" category will be selected.

The classifiers for the individual speaker models can use either supervised or unsupervised training. For supervised training methods (i.e., MLP, NTN, etc.), the classifier for each speaker model is presented with the data for all speakers. Hence, these classifiers are designed with data from both the speaker and the "antispeaker." For unsupervised training (i.e., VQ, Gaussian mixtures, etc.), each speaker model is presented with only the data for that speaker.

A. Nearest Neighbor

The NN, or more general kNN, method [35] can be used to estimate the probability density function (PDF) of the data within a class or itself can be used as a classifier. The simplest kNN classifier is the NN classifier (kNN with $k = 1$), which operates as follows. All training data, i.e., each feature vector along with its corresponding label, are stored. The distances between a given test vector and all training data are computed. The test vector is then assigned the label of the training vector with the smallest distance, i.e., the *nearest neighbor*.

The kNN method ($k > 1$) uses a voting procedure to determine the class of a new vector. This voting procedure is based on the closest k distances or neighbors. For example, given a test vector, its closest k neighbors are found. A vote is then taken to find the most common class among the k nearest neighbors (where k is usually chosen as an odd number to avoid ties). The test vector is then assigned the label of this most common class.

For speaker recognition, typically, a sequence of test vectors is available. For the NN or kNN methods, the labels are found for all test vectors. The score for each class is recorded. For speaker identification, the speaker corresponding to the largest score is selected.

The NN method has been considered for text-independent speaker verification [36] and speaker identification [25]. Higgins [25] used the distances of the nearest neighbors in addition to the class labels. Here, the NN distance measure is defined as

$$\begin{aligned} d(U,R) = &\frac{1}{U}\sum_{u_i \in U} \min_{r_j \in R} |u_i - r_j|^2 \\ &- \frac{1}{U}\sum_{u_i \in U} \min_{u_j \in U, j \neq i} |u_i - u_j|^2 \\ &+ \frac{1}{R}\sum_{r_j \in R} \min_{u_i \in U} |u_i - r_j|^2 \\ &- \frac{1}{R}\sum_{r_i \in R} \min_{r_j \in R, j \neq i} |r_i - r_j|^2 \end{aligned}$$

where $\{u_i\}$ and $\{r_i\}$ are the test and reference feature vectors, and U and R are the number of test and reference feature vectors.

NN classifiers require the storage of all training data and an exhaustive search to find the closest neighbor among all the data. Hence, they are very costly with respect to memory and computation requirements. The NN classifiers considered here only use the labels of the NN and not the distances. The use of distances in addition to the labels should, however, improve performance.

B. Vector Quantization

Clustering algorithms, in general, fall under the category of unsupervised learning, i.e., the class label is not used during training. Clustering will automatically group the training data into its individual modes or classes. A popular iterative approach for computing clusters is called the K-means algorithm. (The K refers to the number of classes or modes.) The K-means algorithm has also been used for coding, where it is typically referred to as VQ [37]. Numerous K-means or VQ algorithms exist, including the Linde-Buzo-Gray (LBG) [38] method, learning vector quantization (LVQ) [39], etc. The results presented here were obtained using the LBG algorithm.

The VQ algorithm can be used as a classifier as follows. First, the feature space spanned by the training vectors $\{x_1, x_2, ..., x_L\}$ is partitioned into a set of mutually exclusive convex regions $\{s_1, s_2, ..., s_N\}$, where $N <= L$. The regions s_i, $1 \leq i \leq N$ each represent a different cluster of data. The centroid of each cluster is computed. The collection of these centroids $\{c_1, c_2, ..., c_N\}$ is typically called a codebook. To classify a test vector, the distance between it and each centroid of the codebook is found. The test vector is assigned to the codebook entry having the smallest distance. Since the classification of a test vector requires a comparison with all vectors in the codebook, this technique is known as full-search vector quantization (FSVQ).

The VQ classifier can be used for speaker recognition [17] as follows. Given the extracted feature vectors from a speaker, a codebook is constructed for that speaker. Typically, 32 or 64 centroids per codebook are used. This process is repeated for all speakers in the population. For speaker identification, the feature vectors from a test utterance are applied to each of the codebooks. For a given codebook, the centroid that is closest to the test vector is found, and the distance to this centroid is accumulated for that codebook. The speaker is selected as corresponding to the codebook with the minimum accumulated distance. For speaker verification [40], the test vectors are only applied to the model for the speaker to be verified. The accumulated minimum distance is computed and normalized to the number of testing vectors. This normalized distance is compared with a threshold to decide if the speaker will be rejected or accepted. Adaptive thresholds that use relative measures to the most similar speakers, or *cohorts*, can also be used. This technique is known as cohort normalization [41], [42].

The VQ classifier has been used for speaker recognition systems with various features including LP coefficients [23], cepstral coefficients [17], and LSP frequencies [43]. Since its introduction in [17], the combination of cepstral coefficients with VQ has provided a good benchmark of peak performance for speaker identification systems.

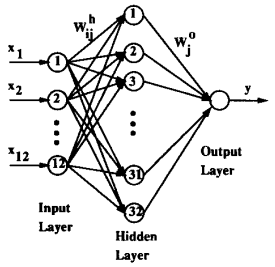


Fig. 3. Multilayer perceptron.

C. Tree-Structured VQ

The VQ classifier has had much success as a classifier for speech-processing applications. However, the search for the closest centroid within a codebook can be cumbersome for a large speaker population. One way to alleviate this problem is to organize the search for the closest centroid in a hierarchical fashion. This can be accomplished with tree-structured VQ.

Tree-structured VQ [44], [45] provides an efficient search algorithm for classifying test vectors. The savings are gained by using a hierarchical decision structure, i.e., a tree, to direct the search for the closest centroid. The tradeoff for this advantage in retrieval time is that the test vector may not be assigned to the optimal cluster.

Speaker recognition can be implemented with tree-structured VQ in the same manner as for the above-described full-search VQ. The difference is that the search time will be reduced at the expense of optimal cluster assignment. Tree-structured VQ has not been considered for speaker recognition previously due to its nonoptimal performance. We shall consider tree-structured VQ in the experimental results section for completeness.

D. Multilayer Perceptron

An alternative to classifiers based on the statistical pattern recognition concept of clustering is supervised learning. One category of supervised classifiers that has recently become popular is neural networks [46]. Neural networks learn complex mappings between inputs and outputs and are particularly useful when the underlying statistics of the considered task are not well understood. Neural networks have been playing an increasing role in speech processing [47] and have only recently been considered for speaker recognition [26]–[30]. The two neural networks considered here are the MLP and NTN.

The MLP is a type of neural network that has grown popular over the past several years. A MLP with one input layer, one hidden layer, and one output layer is shown in Fig. 3.

MLP's are usually trained with an iterative gradient algorithm known as backpropagation [48].

The MLP is convenient to use for problems with limited information regarding characteristics of the input. However, the optimal MLP architecture (number of nodes, hidden layers, etc.) to solve a particular problem must be selected by trial and error, which is a drawback. In addition, the training time required to solve large problems can be excessive, and the algorithm is vulnerable to converging to a local minima instead of the global optimum.

The MLP can be applied to speaker recognition [26] as follows. First, the feature vectors are gathered for all speakers in the population. The feature vectors for one speaker are labeled as "one," and the feature vectors for the remaining speakers are labeled as "zero." An MLP is then trained for that speaker using these feature vectors. The MLP's for all speakers in the population are trained using this method. This distributed labeling scheme is also used for training the decision tree and NTN classifiers, which will discussed shortly. Ideally, test vectors for a specific speaker should have a "one" response for that speaker's MLP, whereas test vectors from different speakers should have a "zero" response. For speaker identification, all test vectors are applied to each MLP, and the outputs of each are accumulated. The speaker is selected as corresponding to the MLP with the maximum accumulated output. For speaker verification, the test vectors are applied to the model of the speaker to be verified. The output is accumulated and normalized to the number of test vectors. If the normalized output exceeds a threshold, the speaker is verified; otherwise, it is rejected.

One problem that is encountered during MLP training for large speaker populations is that the majority of the training vectors have inhibitory (zero) labels, and only a small percentage of the training vectors have excitatory (one) labels. Hence, the classifier tends to learn that "everything" is inhibitory. One method to alleviate this problem is to add noisy duplicates of the excitatory vectors to the training set to even out the distribution of excitatory and inhibitory vectors. However, for a large number of speakers, this results in an unwieldy amount of training data that can hinder the convergence of the classifier. An alternative method to counteract this problem is used here. Rather than increasing the number of excitatory vectors, the number of inhibitory vectors is decreased via compression (VQ). Hence, by representing the inhibitory vectors for each speaker with the corresponding "codebook," the overall number of training vectors is reduced, and the distribution of excitatory and inhibitory vectors is balanced. The experimental results demonstrate this method to be an effective one.

E. Decision Trees

Decision trees are well-known classifiers in the field of statistics. A decision tree represents a collection of rules, which are organized in a hierarchical fashion, that implement a decision structure. Several popular decision trees include Quinlan's C4 [49] and ID3 [50], Breiman's CART [51], and Buntine's Bayes decision tree [52].

In a decision tree, each nonterminal (nonleaf) node of the tree represents a decision, and each terminal (leaf) node corresponds to a class. The leaves represent exclusive partitions of the input data. A test vector will be evaluated at each node and directed to one of two subsequent nodes based on a decision. This decision process continues until the test vector comes to a leaf, at which point, it will be assigned the class label of that leaf.

Decision trees, in general, only consider one element at a time when making a decision. This constrains the partitioning of feature space to using discriminants that are perpendicular to the feature axes. This can be a severe limitation for problems that require more flexibility in the discriminant positioning, i.e., a diagonal discriminant. The ID3 and C4 decision trees are subject to this constraint. The CART decision tree allows for discriminants that are a function of all feature elements. However, the algorithm used to find these discriminants is heuristic and involves an exhaustive search; hence, it is a computationally unappealing alternative.

The architecture of a decision tree is found by rule induction [51], [53] and can be determined recursively as follows. If all of the data at a node belongs to one class, then designate this node as a leaf, and assign to it the label of that class. Otherwise, select the feature that will provide the largest information gain for a subsequent data partitioning. A decision is implemented based on this feature and the data is split accordingly. Each partition of the data is sent to a new node, and the above algorithm is repeated. A decision tree grown with this algorithm will completely classify the training data. However, it is well-known that classifiers that have 100% performance on the training data are typically overtrained and will perform suboptimally on the test data. This problem can be alleviated by *pruning* [51] the tree. Pruning consists of removing the subtrees that may have been grown excessively to classify a small portion of the training data. Here, performance on the training data will be compromised to improve performance on the testing data.

Decision trees have some advantages over MLP's, which include self-organizing architectures that do not have to be specified *a priori* as with MLP's. On the other hand, MLP's are advantageous over decision trees in that the partitioning of feature space is not constrained to discriminants that are perpendicular to the feature axes. The performance of decision trees and MLP's have been compared and evaluated for speech processing tasks, such as vowel recognition [54]. Here, it was found that the MLP (with the proper architecture selection) always performed at least as well as decision trees.

A decision tree can be used for speaker recognition as follows. First, the feature vectors are obtained from the training data for all speakers. The data is then labeled in a identical fashion to the method previously discussed for MLP's. A binary decision tree is trained for each speaker. The leaves of the binary decision tree will contain the class label, where a one will correspond to the speaker and a zero will correspond to "not the speaker," or "antispeaker." In addition to the class labels, the leaves of the decision tree also contain a probability associated with the label. For speaker identification, all feature vectors for the test utterance are applied to each decision tree. The likelihood measure of each speaker based on the decision tree probabilities is used to determine the speaker.

IV. Neural Tree Network

The neural tree network (NTN) [55] is a hierarchical classifier that combines the properties of feedforward neural networks and a decision tree [51] type structure. Each node of the tree consists of a single layer perceptron (SLP). The SLP uses all feature elements for the decision, and hence, the NTN is not constrained to perpendicular discriminant boundaries as are standard decision trees. The architecture of the NTN is determined during training; thus, it is self-organizing. This is in contrast to MLP's where the architecture must be specified prior to training.

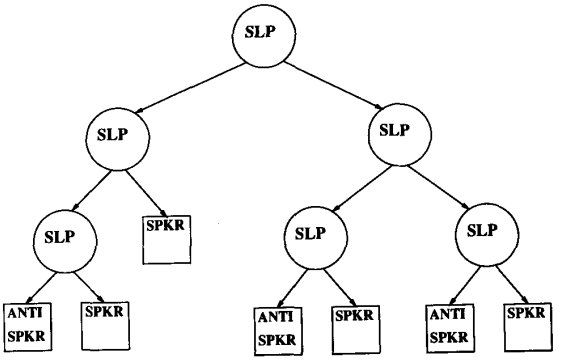


Fig. 4. Neural tree network.

The NTN uses a tree architecture to implement a sequential linear decision strategy [56]. Each node at every level of the NTN divides the input vectors into a number of exclusive subsets of the training data. The leaf nodes of the NTN partition the feature space into homogeneous subsets, i.e., a single class at each leaf node. The NTN is recursively trained as follows. Given a set of training data at a particular node, if all data within that node belongs to the same class, the node becomes a leaf. Otherwise, the data is split into several subsets, which become the children of this node. This procedure is repeated until all the data is completely uniform at the leaf nodes. Hence, a trained NTN will have 100% performance on the training set. However, the resulting NTN may not have optimal generalization due to overtraining. As with decision trees, the generalization can be improved, *pruning* the NTN [5]. This will remove the subtrees that overfit the data.

For speaker recognition, a binary NTN is grown for each speaker. The feature vectors are extracted from the training utterances for each speaker in the training set and labeled according to the distributed labeling scheme described in Section III-D. All training vectors are then applied to the NTN. The NTN is recursively grown until it completely classifies all of the patterns in the training set. Note that this is the training algorithm for the standard NTN. The training algorithm for the MNTN, which uses forward pruning, will be discussed shortly. A typical binary NTN grown for the speaker recognition task is illustrated in Fig. 4.

Here, the leaves labeled "spkr" correspond to the "one" class, and leaves labeled "antispkr" correspond to the "zero" class. Note in Fig. 4 that the training data for a given class does not necessarily fall into the same leaf but more likely is dispersed among several leaves.

A trained NTN can be used for speaker recognition as follows. For speaker identification, the test vectors are applied to all speaker NTN's, and the percent of vectors belonging

to each model is computed. If the problem is closed set, the speaker is selected as corresponding to the model with the largest percent. If the problem is open set, the largest percent is compared with a threshold. If the percent is above the threshold, the speaker is selected; otherwise, the classifier outputs an "out of set" classification. For speaker verification, all feature vectors from the test utterance are input to the NTN belonging to the speaker to be verified. Each vector will be assigned a label corresponding to whether or not it belongs to the speaker. If the percent of test vectors belonging to the speaker exceeds a predetermined threshold, the speaker is verified.

A related hierarchical classifier scheme was proposed for speaker identification in [30]. This method consists of using a tree structure of binary classifiers to implement an N-way classification task. The method proposed here is different in that N tree-structured binary classifiers are used, as opposed to the tree structure of $(N-1)N/2$ classifiers in [30].

Several extensions to the NTN have been developed that include 1) hyperplane adjustment for on-line training [57] and 2) data relabeling [58]. Two new enhancements are proposed here that improve the NTN training and selection of the final architecture. These enhancements consist of a new learning rule and pruning algorithm and are described in the following sections.

A. Minimum Classification Error

The splitting algorithm of the modified NTN is based on discriminant learning [59]. Discriminant learning uses a cost function that minimizes the classification error. This is contrasted with the more traditional learning algorithms that minimize a norm of the approximation error. At each node, the NTN computes the inner product of a weight vector w and an input feature vector x, which should be approximately equal to the the output label $y \in \{0,1\}$. Learning algorithms typically minimize a norm of the error $\epsilon = (y - < w, x >)$, such as the L_2 or L_1 norm. Discriminant learning instead minimizes the classification error. This is explained as follows.

For an M-class NTN, the discriminant learning approach first defines a misclassification measure $d(x)$ as

$$d(x) = - < w_i, x > + \left\{ \frac{1}{M-1} \sum_{j \neq i} (< w_j, x >)^n \right\}^{\frac{1}{n}} \quad (3)$$

where n is a predetermined smoothing constant. The vector x will be assigned to the class corresponding to the weight vector w_k having the maximum inner product with x. Hence, in (3), if x belongs to class i, then $d(x)$ should be negative, and if x does not belong to class i, $d(x)$ should be positive. The misclassification measure $d(x)$ is then applied to a sigmoid to yield

$$g[d(x)] = \frac{1}{1 + e^{-d(x)}}. \quad (4)$$

The sigmoid transforms misclassifications to a quantity roughly equal to one and correct classifications to a quantity roughly equal equal to zero. The total error for all input vectors can be found as

$$E = \sum_{k=1}^{N} g[d(x_k)] \quad (5)$$

where N is the number of inputs. The quantity E is approximately equal to the number of misclassifications. Hence, minimizing E will tend to minimize the classification error. It is noted that for binary NTN's, the weight updates for discriminant learning and the L_1 norm are equivalent [60].

B. Forward Pruning

The pruning algorithm presented in [5] minimizes a Lagrangian cost function to improve the performance of the tree. The cost function consists of a weighted combination of two minimization criteria. These are 1) minimizing the number of misclassifications on the training data and 2) minimizing the number of leaf nodes. First, the tree is grown during training to obtain 100% classification on the training set. The pruning procedure searches through different subtrees formed by pruning certain branches as determined by the Lagrangian cost function. An independent pruning set of data is then used to evaluate each of the subtrees. The subtree having the best performance on the pruning set is selected as the optimal pruned tree. Since this method requires first growing the tree followed by pruning back the branches, we call it a *backward* pruning technique.

The backward pruning algorithm has some disadvantages. First, the backward pruning method assumes that the data used to prune the tree is fully representative of the feature space. Hence, data that is biased or does not represent all classes will cause the selection of a nonoptimal pruned tree. Second, for very large trees, such as those encountered in speaker recognition applications, the number of subtrees that must be constructed and evaluated becomes excessive. This paper considers an alternative pruning strategy known as *forward* pruning.

Forward pruning consists of pruning the tree while it is being grown. This is accomplished by halting the growth of a tree beyond a prespecified level. Since the leaves at this truncated level are likely to have data from several classes, a vote is taken to determine the most frequent class at this leaf. This leaf is then assigned the label of the vote winner. In addition, a confidence measure is computed at each leaf. This measure consists of the ratio of the vote winner to the number of votes from the other classes. Hence, vote winners having large majorities will result in a high confidence, whereas marginal vote victories will have a low confidence. This confidence measure can be used in conjunction with the class label in classification tasks, such as speaker recognition.

C. Speaker Recognition Using the MNTN

The MNTN can be used for speaker recognition in a similar fashion to the original NTN, as described in Section IV. The main difference is that the MNTN uses the confidence measure at each leaf in addition to the class label. Given a sequence of feature vectors from a test utterance, the confidence measure of all vectors classified as "speaker" is

accumulated. Similarly, the confidence measure for all vectors classified as "antispeaker" is accumulated. The speaker likelihood is then computed as the ratio of the accumulated speaker confidence measure to the sum of the accumulated speaker and antispeaker confidence measures. For example, consider an observed sequence of feature vectors x as applied to the model for the jth speaker S_j. The speaker likelihood for the NTN is computed as

$$P_{NTN}(x|S_j) = \frac{M}{N+M} \tag{6}$$

where M is the number of vectors classified as the speaker, and N is the number of vectors classified as the antispeaker. The corresponding score for the MNTN is computed as

$$P_{MNTN}(x|S_j) = \frac{\sum_{i=1}^{M} c_i^1}{\sum_{j=1}^{N} c_j^0 + \sum_{i=1}^{M} c_i^1} \tag{7}$$

where c^1 and c^0 are the confidence measures, given that a vector was classified as the speaker or antispeaker, respectively. The addition of confidence measures yields significant improvement over the scoring methods usually used for pruned NTN's.

V. Experimental Results

Several speaker recognition experiments were performed to evaluate the various classifiers. These experiments include closed-set speaker identification, speaker verification, and open-set speaker identification. The following section provides a discussion of the database and preprocessing steps. This section is followed by the descriptions and results of the experiments.

A. The Database

The database for the experiments reported in this paper is a subset of the DARPA TIMIT database. This set represents 38 speakers of the same (New England) dialect. The set of speakers includes 24 males and 14 females.

The preprocessing of the TIMIT speech data consists of several steps. First, the speech is downsampled from 16 to 8 KHz sampling frequency. The downsampling is performed to obtain a toll quality signal. The speech data is processed by a silence-removing algorithm followed by the application of a pre-emphasis filter $H(z) = 1 - 0.95z^{-1}$. A 30-ms Hamming window is applied to the speech every 10 ms. A 12th-order linear predictive (LP) analysis is performed for each speech frame. The features consist of the 12 cepstral coefficients derived from this LP polynomial.

There are ten utterances for each speaker in the selected set. Five of the utterances are concatenated and used for training. The remaining five sentences are used individually for testing. The duration of the training data ranges from 7 to 13 s per speaker, and the duration of each test utterance ranges from 0.7 to 3.2 s.

B. Closed-Set Speaker Identification

The first set of experiments consists of evaluating various classifiers for closed-set speaker identification using 5, 10, and 20 speakers from the original speaker corpus. The classifiers based on unsupervised training that are considered for this experiment are the FSVQ and TSVQ classifiers. The codebook sizes are varied from 16 to 128 for both classifiers. The results are summarized in Tables I and II. Note, as expected, that the nonoptimal cluster assignment for the tree-structured VQ results in a significant degradation with regard to full-search VQ.

TABLE I
Full-Search VQ Closed-Set Speaker Identification Performance

Codebook Size	5 Speakers	10 Speakers	20 Speakers
16	100%	98%	90%
32	100%	98%	92%
64	100%	98%	95%
128	100%	98%	96%

TABLE II
Tree-Structured VQ Closed-Set Speaker Identification Performance

Codebook Size	5 Speakers	10 Speakers	20 Speakers
16	96%	92%	83%
32	100%	92%	90%
64	100%	96%	90%
128	96%	94%	88%

TABLE III
KNN Closed-Set Speaker Identification Performance

Nearest Neighbors	5 Speakers	10 Speakers	20 Speakers
1	96%	96%	85%
3	84%	90%	85%
5	96%	96%	89%
7	100%	96%	92%

The next classifier evaluated for this experiment is the kNN method [35]. The number of nearest neighbors k is varied from one to seven. The results of the kNN method are summarized in Table III. In Table III, it can be seen that the performance as a function of the number of NN's is not monotonic. Hence, no immediate conclusions can be drawn for how to select k. In general, however, the results suggest that the larger the speaker population, the larger k should be. This is logical since for large populations, there will be more confusion among the nearest neighbors. Here, a larger vote should give a more reliable statistic for the NN region.

The next classifier considered for closed-set speaker identification is the MLP. The MLP is trained using the back-propagation algorithm [48] for architectures having 16, 32, and 64 hidden nodes (within one hidden layer). It has been previously shown [26] that using more than one hidden layer does not improve performance. Note that the data for each antispeaker is compressed to 100 vectors prior to constructing the training set for a given speaker. The results for the MLP classifier are summarized in Table IV. One would expect that increasing the number for hidden nodes would improve performance. However, increasing the number of hidden nodes results in a more complicated partitioning of feature space that is more subject to "getting stuck" in a local minima. It is suspected that this phenomena is responsible for the fall off in performance seen in Table IV as we increase the number of hidden nodes.

TABLE IV
MLP CLOSED-SET SPEAKER IDENTIFICATION PERFORMANCE

Hidden Nodes	5 Speakers	10 Speakers	20 Speakers
16	96%	90%	90%
32	96%	90%	82%
64	88%	94%	85%

TABLE V
DECISION TREE CLOSED-SET SPEAKER IDENTIFICATION PERFORMANCE

Decision Tree	5 Speakers	10 Speakers	20 Speakers
ID3	88%	88%	79%
C4	92%	84%	73%
CART	80%	76%	*
BAYES	92%	92%	83%

TABLE VI
NTN AND MNTN CLOSED-SET SPEAKER IDENTIFICATION PERFORMANCE

Pruning Level	5 Speakers (ntn/mntn)	10 Speakers (ntn/mntn)	20 Speakers (ntn/mntn)
4	80/92%	66/88%	67/75%
5	88/96%	84/90%	76/87%
6	96/100%	92/94%	82/93%
7	92/96%	92/98%	91/96%
8	92/92%	90/96%	89/94%
no pruning	92/92%	90/90%	89/89%

Next, several decision trees are applied to the closed-set speaker identification task. The decision trees used in this experiment are C4 [49], ID3 [50], CART [51], and a Bayesian decision tree [52]. The results are summarized in Table V. Here, the Bayesian decision tree performed the best overall. The * denotes that the CART tree was unable to grow for the 20-speaker experiment due to computer memory limitations.

The NTN and MNTN are next applied to the closed-set speaker identification experiments. The performance of the NTN and MNTN is evaluated for different pruning levels. These results are summarized in Table VI, where the MNTN is found to consistently outperform the NTN. In Table VI, it can be seen that pruning the tree can improve generalization for all cases. However, the optimal pruning level occurs at different levels for different size speaker populations. From this result, one can conjecture that the optimal pruning level is a function of the *confusion* of the data.

Note that the MNTN's pruned at the sixth or seventh level achieve the performance of full-search VQ and outperform tree-structured VQ. Hence, with the proper selection of a pruning level, the MNTN can provide the performance of full-search VQ with the search efficiency of tree-structured VQ.

The results for the best overall configuration of each classifier are tabulated in Table VII. Here, it can be seen that the full-search VQ classifier and MNTN provide the best overall performance for the closed-set speaker identification problem.

The next experiment evaluates the MNTN and full-search VQ classifiers for training data of variable duration. The previous experiments use five concatenated sentences for training. This experiment evaluates the performance of the MNTN and VQ classifiers for the data from the first sentence, the data for the first two sentences, etc., up to the first five sentences. Each sentence is, on average, 2 s. The performance versus training data duration for the VQ classifier (128 codebook entries) and MNTN (seven levels) is shown in Fig. 5 for ten speakers and Fig. 6 for 20 speakers.

Here, it can be seen in both cases that the MNTN performs better for short training utterances than the VQ classifier.

TABLE VII
CLOSED-SET SPEAKER IDENTIFICATION PERFORMANCE

Classifier	5 Speakers	10 Speakers	20 Speakers
FSVQ (128)	100%	98%	96%
TSVQ (64)	100%	94%	88%
MNTN (7 level)	96%	98%	96%
MLP (16)	96%	90%	90%
ID3	88%	88%	79%
CART	80%	76%	*
C4	92%	84%	73%
BAYES	92%	92%	83%

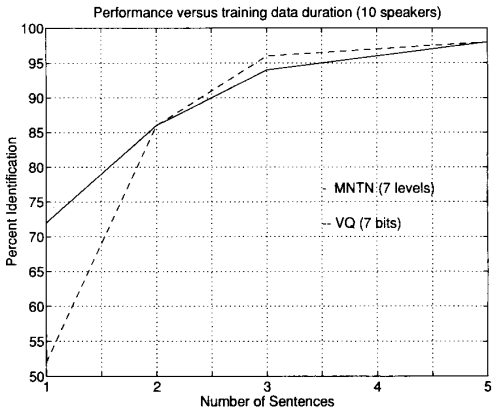


Fig. 5. VQ and MNTN performance versus training duration (ten speakers).

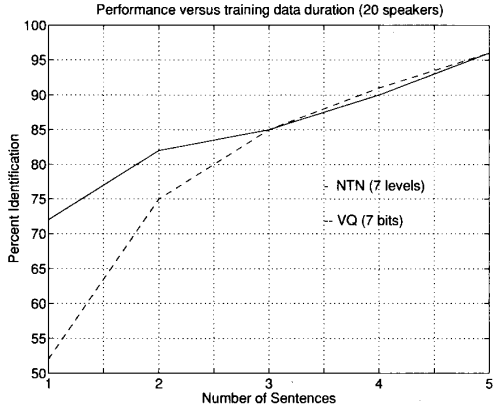


Fig. 6. VQ and MNTN performance versus training duration (20 speakers).

C. Speaker Verification

The full-search VQ classifier and MNTN are evaluated next for speaker verification. The first speaker verification experiment consists of ten speakers and ten imposters (i.e.,

people not used in the training set). Both sets have five males and five females. Each MNTN uses seven levels and is trained with the data for all ten speakers. Scores for the MNTN are found as the ratio of accumulated speaker confidence to accumulated imposter confidence. The full-search VQ classifier is evaluated for a codebook size of 128 entries. Here, the scores are found as the accumulated distance measure normalized to the number of test vectors.

Performance of the VQ classifier for speaker verification can be improved by using a technique known as cohort normalization [42]. Traditional speaker verification systems will accept a speaker if

$$p(X|I) > T(I) \tag{8}$$

where $p(X|I)$is the likelihood that the sequence of feature vectors X was generated by speaker I and $T(I)$ is the corresponding likelihood threshold. Instead of using the fixed threshold criteria in (8), an adaptive threshold can be used via the likelihood measure

$$P(X|I) > P(X|\bar{I}). \tag{9}$$

Here, a speaker is verified if the likelihood $P(X|I)$ is greater than the likelihood that the feature vectors X were generated by a speaker other than I. The problem now is how to estimate the "antispeaker" likelihood $p(X|\bar{I})$. One technique is to estimate $P(X|\bar{I})$ based on the closest speaker models to I, which are denoted as I's *cohorts* [42]. When verifying a speaker, the feature vectors will be applied to the model for speaker I and, in addition, the models for I's cohorts. The score for speaker I is then normalized based on some metric of the cohort scores. This metric can consist of a maximum, minimum, average, etc., depending on the classifier used.

Speaker verification experiments are performed here using both the fixed threshold technique in (6) and the *cohort normalized* score represented by (7). For the VQ method, a speaker's score is normalized by the *minimum* score among the cohorts. For the MNTN, a speaker's score is normalized by the *average* score among the cohorts. These metrics are found to perform best for each respective classifier. In addition, for a given speaker, the cohort set consists of the remaining nine speakers in the enrolled population. In practical applications, only a subset of the enrolled speakers would be selected as cohorts. Instead, all cohorts are used here to get a measure of the best achievable performance.

The threshold for the VQ accumulated distortion and MNTN accumulated confidence is varied from the point of 0% false acceptance to 0% false rejection to yield the operating curves shown in Fig. 7. Note that all operating curves presented in this section for speaker verification and open-set speaker identification represent the posterior performance of the classifiers, given the speaker and imposter scores. In Fig. 7, it can be seen that the MNTN and VQ classifiers are both improved by the cohort normalized scores. The equal error rates for the MNTN and VQ classifier are summarized in Table VIII.

The second speaker verification experiment consists of 20 speakers and 18 imposters. The set of 20 speakers consists of ten males and ten females. The set of 18 imposters consists of

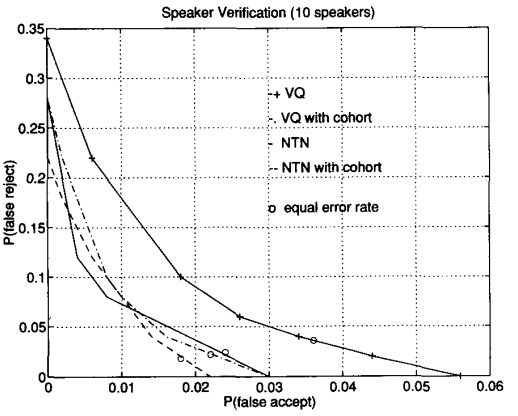


Fig. 7. Speaker verification (ten speakers).

TABLE VIII
EQUAL ERROR RATES FOR SPEAKER VERIFICATION (TEN SPEAKERS)

	VQ	MNTN
without cohort	3.6%	2.4%
with cohort	2.2%	1.8%

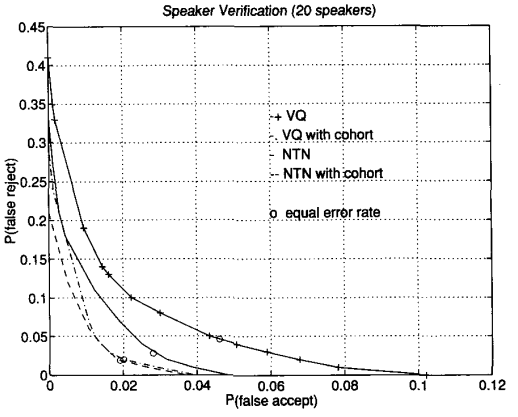


Fig. 8. Speaker verification (20 speakers).

TABLE IX
EQUAL ERROR RATES FOR SPEAKER VERIFICATION (20 SPEAKERS)

	VQ	MNTN
without cohort	4.6%	2.8%
with cohort	2.0%	1.9%

14 males and four females. For a given speaker, the cohort set for the MNTN and VQ classifiers consists of the remaining 19 speakers in the enrolled population. The operating curves for the MNTN and VQ classifiers are shown in Fig. 8. Again, the best performance is achieved with the cohort normalized scores. The equal error rates for the MNTN and VQ classifiers are summarized in Table IX. For both experiments (10 and 20 speakers), the MNTN provides better performance than the VQ classifier, both with and without cohort normalization, for most of the operating curve.

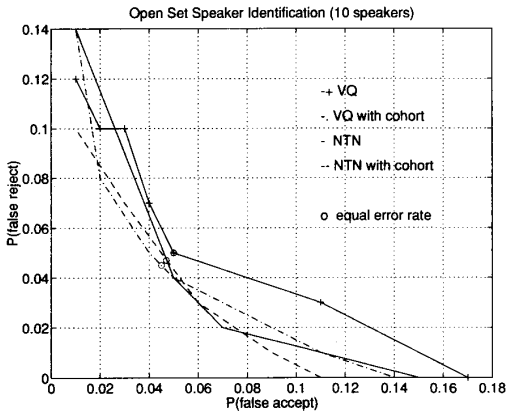


Fig. 9. Open-set speaker identification (ten speakers).

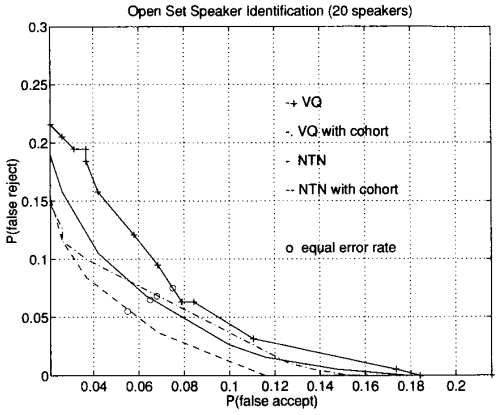


Fig. 10. Open-set speaker identification (20 speakers).

TABLE X
EQUAL ERROR RATES FOR OPEN-SET IDENTIFICATION (TEN SPEAKERS)

	VQ	MNTN
without cohort	5.0%	4.7%
with cohort	4.5%	4.7%

TABLE XI
EQUAL ERROR RATES FOR OPEN-SET IDENTIFICATION (20 SPEAKERS)

	VQ	MNTN
without cohort	7.5%	6.5%
with cohort	6.8%	5.5%

D. Open Set Speaker Identification

The MNTN and VQ classifiers are also evaluated for open-set speaker identification. The open-set speaker identification experiments use the same data set and classifier training procedure as the speaker verification experiments. Cohort normalized scores are also considered here. For open-set speaker identification, not only must the speaker have the maximum confidence measure (or minimum distance), but this measure must also be subject to a threshold. If it does not satisfy the threshold, it is scored as an error, i.e., a false reject. Similarly, all imposters must be checked for identification. The minimum imposter distance or maximum score must not satisfy the threshold. If it does, then the imposter will incorrectly be identified as someone within the set, i.e., a false accept. The operating curve for a ten-speaker open-set identification task is shown in Fig. 9. The equal error rates for the MNTN and VQ classifiers are summarized in Table X.

The open-set speaker identification experiment is repeated for the data set comprised of 20 speakers and 18 imposters. The operating curve for this experiment is shown in Fig. 10. The equal error rates for the MNTN and VQ classifiers are summarized in Table XI.

In general, the cohort normalization provides a larger performance improvement for the VQ classifier than for the MNTN. This is due to the fact that the MNTN uses "antispeaker" data for training; hence, the likelihood measure is, in some sense, built into the classifier. Nonetheless, the cohort normalization also improves the performance of the MNTN.

VI. CONCLUSIONS

A comparison of various classifiers based on statistical pattern recognition and neural networks has been performed for text-independent speaker recognition. Additionally, a new classifier called the modified NTN is examined for this application. The performance of the new classifier is evaluated for several speaker recognition experiments using various-sized speaker sets from a 38 speaker corpus. The features used to evaluate the classifiers are the LP-derived cepstrum. The classifier is compared with full-search and tree-structured VQ classifiers, NN classifiers, multilayer perceptrons, and decision trees. The classifiers are first evaluated for a closed-set speaker identification experiment with 5 to 20 speakers. The full-search VQ and MNTN classifiers both demonstrate a 4% error rate for the 20-speaker experiment. For a speaker verification experiment with ten speakers and 1ten imposters, the best results are obtained with cohort normalized scores, for which the MNTN and full-search VQ classifiers provide equal error rates of 1.8 and 2.2%. Similarly, for a speaker-verification experiment with 20 speakers and 18 imposters, the MNTN and full-search classifiers provide equal error rates of 1.9 and 2%, respectively, for the cohort normalized scoring. The MNTN and the full-search VQ classifier are also compared for open-set speaker identification. For an experiment with ten speakers, the MNTN provides an error rate of 4.7% as compared with the 4.5% equal error rate obtained by the full-search VQ classifier. For the experiment with 20 speakers and 18 imposters, the MNTN and full-search VQ classifiers yield equal error rates of 5.5 and 6.5%, respectively. In addition to performance advantages for problems requiring imposter rejection, the MNTN also demonstrates a logarithmic saving in retrieval time over that of the full-search VQ classifier. This computational advantage can be obtained by using tree-structured VQ, although tree-structured VQ will reduce the performance with respect to full-search VQ.

ACKNOWLEDGMENT

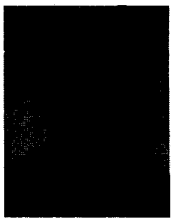
The authors would like to thank J. Couples, J. Flanagan, J. Grieco, and A. Sankar for their useful comments and

suggestions, as well as J. Huang, D. Naik, R. Ramachandran, S. Sivaprasad, and W. Wang for their assistance with the simulations and reviewal of the manuscript. The authors would also like to thank the reviewers for their useful comments and suggestions. The decision tree simulations utilized the IND software package developed by W. Buntine of NASA.

REFERENCES

[1] B. Atal, "Automatic recognition of speakers from their voices," *Proc. IEEE*, vol. 64, pp. 460–475, Apr. 1976.
[2] A. Rosenberg, "Automatic speaker recognition: A review," *Proc. IEEE*, vol. 64, pp. 475–487, Apr. 1976.
[3] G. Doddington, "Speaker recognition—Identifying people by their voices," *Proc. IEEE*, vol. 73, pp. 1651–1664, 1985.
[4] D. O'Shaughnessy, "Speaker recognition," *IEEE ASSP Mag.*, pp. 4–17, Oct. 1986.
[5] A. Sankar and R. J. Mammone, "Growing and pruning neural tree networks," *IEEE Trans. Comput.*, vol. C-42, pp. 221–229, Mar. 1993.
[6] S. Pruzansky, "Pattern-matching procedure for automatic talker recognition," *J. Acoust. Soc. Amer.*, vol. 35, pp. 354–358, Mar. 1963.
[7] P. D. Bricker *et al.*, "Statistical techniques for talker identification," *Bell Syst. Techn. J.*, vol. 50, pp. 1427–1454, Apr. 1971.
[8] B. S. Atal, "Automatic speaker recognition based on pitch contours," Ph.D. thesis, Polytechnic Inst., Brooklyn, NY, June 1968.
[9] G. Doddington, "A method of speaker verification," *J. Acoust. Soc. Amer.*, vol. 49, p. 139(A), Jan. 1971.
[10] C. Lummis, "Speaker verification by computer using speech intensity for temporal registration," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 80–89, Jan. 1973.
[11] S. Furui and F. Itakura, "Talker recognition by statistical features of speech sounds," *Electron. Commun.*, vol. 56-A, pp. 62–71, 1973.
[12] M. R. Sambur, "Speaker recognition and verification using linear prediction analysis," Ph.D. thesis, Mass. Inst. of Technol., Sept. 1972.
[13] J. J. Wolf, "Efficient acoustic parameters for speaker recognition," *J. Acoust. Soc. Amer.*, vol. 51, pp. 2044–2055, June 1972.
[14] M. R. Sambur, "Selection of acoustic features for speaker identification," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-23, pp. 176–182, Apr. 1975.
[15] B. Atal, "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification," *J. Acoust. Soc. Amer.*, vol. 55, pp. 1304–1312, June 1974.
[16] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-29, pp. 254–272, Apr. 1981.
[17] F. K. Soong, A. E. Rosenberg, L. R. Rabiner, and B. H. Juang, "A vector quantization approach to speaker recognition," in *Proc. ICASSP*, pp. 387–390, 1985.
[18] G. Velius, "Variants of cepstrum based speaker identity verification," in *Proc. ICASSP*, pp. 583–586, 1988.
[19] A. V. Oppenheim and R. W. Schafer, "Homomorphic analysis of speech,"*IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 221–226, June 1968.
[20] F. K. Soong and A. E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-36, pp. 871–879, June 1988.
[21] K. Assaleh, R. J. Mammone, and J. L. Flanagan, "Robust features for speaker recognition," in *Proc. Speech Res. Symp.* (Baltimore, MD), Johns Hopkins University, June 1993.
[22] W. Ren-hua, H. Lin-shen, and H. Fujisaki, "A weighted distance measure based on the fine structure of feature space: Appliction to speaker recognition," in *Proc. ICASSP*, 1990, pp. 273–276.
[23] K. Li and E. Wrench, "An approach to text-independent speaker recognition with short utterances," in *Proc. ICASSP*, 1983, pp. 555–558.
[24] J. B. Attilli, "A TMS320c20 based real time, text-independent automatic speaker verification system," in *Proc. ICASSP*, 1988.
[25] A. L. Higgins, L. G. Bahler, and J. E. Porter, "Voice identification using nearest-neighbor distance measure," in *Proc. ICASSP*, 1993, pp. 375–378.
[26] J. Oglesby and J. S. Mason, "Optimization of neural models for speaker identification," in *Proc. ICASSP*, 1990, pp. 261–264.
[27] Y. Bennani and P. Gallinari, "On the use of TDNN-extracted features information in talker identification," in *Proc. ICASSP*, 1991, pp. 385–388.
[28] J. Oglesby and J. S. Mason, "Radial basis function networks for speaker recognition," in *Proc. ICASSP*, 1991, pp. 393–396.
[29] Y. Bennani and P. Gallinari, "A connectionist approach for speaker identification," in *Proc. ICASSP*, 1990, pp. 265–268.
[30] L. Rudasi and S. A. Zahorian, "Text-independent talker identification with neural networks," in *Proc. ICASSP*, 1991, pp. 389–392.
[31] R. C. Rose and D. A. Reynolds, "Text independent speaker identification using automatic acoustic segmentation," in *Proc. ICASSP*, 1990, pp. 293–296.
[32] M. Savic and S. K. Gupta, "Variable parameter speaker verification system based on hidden Markov modeling," in *Proc. ICASSP*, 1990, pp. 281–284.
[33] A. E. Rosenberg, C. H. Lee, and S. Gokeen, "Connected word talker recognition using whole word hidden Markov models," in *Proc. ICASSP*, 1991, pp. 381–384.
[34] N. Z. Tishby, "On the application of mixture AR hidden Markov models to text independent speaker recognition," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-39, pp. 563–569, Mar. 1991.
[35] K. Fukunaga, *An Introduction to Statistical Pattern Recognition.* San Diego, CA: Academic, 1990.
[36] J. P. Campbell, "Features and measures for speaker recognition," Ph.D. thesis, Oklahoma State Univ., Dec. 1992.
[37] R. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4–29, Jan. 1984.
[38] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, 1981.
[39] T. Kohonen, *Self-Organization and Associative Memory.* New York: Springer-Verlag, 1989, 3rd ed.
[40] A. E. Rosenberg and F. K. Soong, "Evaluation of a vector quantization talker recognition system in text independent and text dependent modes," *Comput. Speech Language*, vol. 22, pp. 143–157, 1987.
[41] A. Higgins and L. Bahler, "Text-independent speaker verification by discriminator counting," in *Proc. ICASSP*, 1991, pp. 405–408.
[42] A. E. Rosenberg, J. Delong, C. H. Lee, B. H. Juang, and F. K. Soong, "The use of cohort normalized scores for speaker recognition," in *Proc. ICSLP-92*, Oct. 1992, pp. 599–602.
[43] C. Liu, M. Lin, W. Wang, and H. Wang, "Study of line spectral pair frequencies for speaker recognition," in *Proc. ICASSP*, 1990, pp. 277–280.
[44] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer, 1992.
[45] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization for speech coding," *Proc. IEEE*, vol. 73, pp. 1551–1587, Nov. 1985.
[46] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4–22, Apr. 1987.
[47] D. P. Morgan and C. L. Scofield, *Neural Networks and Speech Processing.* Norwell, MA: Kluwer, 1991.
[48] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing.* Cambridge, MA: MIT Press, 1986.
[49] J. Quinlan, *Simplifying Decision Trees in Knowledge Acquisition for Knowledge-Based Systems* (G. Gaines and J. Boose, Ed.). London, Academic, 1988.
[50] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
[51] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* Wadsworth: 1984.
[52] W. Buntine, "Learning classification trees," *Stat. Comput.*, vol. 2, pp. 63–73, 1992.
[53] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn.* San Mateo, CA: Morgan Kaufmann, 1991.
[54] L. Atlas *et al.*, "A performance comparison of trained multilayer perceptrons and trained classification trees," *Proc. IEEE*, vol. 78, pp. 1614–1619, Oct. 1990.
[55] A. Sankar and R. J. Mammone, *Neural Tree Networks in Neural Networks: Theory and Applications* (R. J. Mammone, Ed.). San Diego, CA: Academic, 1991.
[56] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.
[57] A. Agrawal and R. J. Mammone, "An on-line training algorithm to overcome catastrophic forgetting," in *Proc. ANNIE* (St. Louis, MO), 1992.
[58] W. Wu and R. J. Mammone, "The relabeling exchange method (REM) for training neural networks," in *Neural Networks for Speech and Vision Processing* (R. J. Mammone, Ed.). Chapman and Hall, 1993, to be published.

[59] S. Katagiri, B. H Juang, and A. Biem, "Discriminative feature extraction," in *Neural Networks for Speech and Vision Processing* (R. J. Mammone, Ed.). Chapman and Hall, 1993, to be published.
[60] A. Sankar, Private communication, 1993.

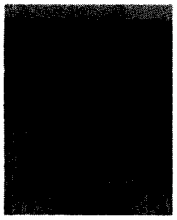


Kevin R. Farrell (S'86–M'93) received the B.S., M.S., and Ph.D. degrees in electrical engineering in 1988, 1991, and 1993, all from Rutgers University, Piscataway, NJ.

Since August 1991, he has been with the CAIP Center of Rutgers University, where he is currently a Research Professor. He has also consulted with AT&T Bell Laboraties for problems in natural language acquistion. His current research interests lie in pattern recognition with applications in speech processing.

Dr. Farrell was a Federal Aviation Administration (FAA) Fellow during 1992–1993. He has contributed several chapters to texts in signal processing and neural networks.

Richard J. Mammone (S'75–M'81–SM'86) for a photograph and biography, please see page 114 of this TRANSACTIONS.



Khaled T. Assaleh (S'91–M'93) received the B.S. degree in electrical engineering from the University of Jordan, Amman, Jordan, in 1988. He received the M.S. degree from Monmouth College, West Long Branch, NJ, in 1990 and the Ph.D. degree from Rutgers University, Piscataway, NJ, in 1993, both in electrical engineering.

He is currently a Research Professor at the CAIP Center of Rutgers University. During 1989–1990, he worked at Bell Communications Research on low bit rate video coding. His current research interests are in the areas of speech and speaker recognition. He has written numerous publications in the field of signal processing.