Neuro-Fuzzy and Soft Computing for Speaker Recognition

Jyh-Shing Roger Jang and Jiuann-Jye Chen
CS Department, Tsing Hua University, Hsinchu, Taiwan
Email: {jang,mr844339}@cs.nthu.edu.tw

Keywords: Random search, fuzzy c-means clustering, fuzzy logic, soft computing, speaker recognition, digital signal processing.

Abstract

This paper describes how techniques from the discipline of neuro-fuzzy and soft computing techniques can be used, in conjunction with methodologies from pattern recognition and digital signal processing, to effectively perform speech data classification. In particular, we have applied the proposed method to automatic speaker recognition and achieved satisfactory results.

1. Introduction

This paper describes our experiments of using neuro-fuzzy and soft computing techniques [5] to a challenging real-world problem: automatic speaker recognition (ASR). Given a speech input, the objective of ASR is to output the identity of the person most likely to have spoken. ASR is a difficult problem in pattern recognition. It involves typically a huge amount of data and we need to apply digital signal processing techniques to down-size the data dimension and extract relevant features for further processing of data classification or discriminant analysis. For such a difficult problem, a single approach is usually not enough and we need a collection of various methodologies to complement each other to accomplish the task. Hence the techniques of neuro-fuzzy and soft computing, are introduced in order to solve the computation-intensive problem of ASR.

2. Automatic Speech Recognition

The task of text-independent automatic speaker recognition (ASR) is to determine the identity of a speaker by machine. By text-independent, we mean that the recognition procedure should work for any text uttered by the speaker. This is different problem than text-dependent recognition, where the text comes from a predefined set. ASR may be viewed as a complement to speech recognition, where the latter attempts to decode the linguistic message (or text) underlying an utterance, rather than the identity of the speaker. For ASR, the speech signal must be processed to extract measures of speaker variability instead of segmental features.

Speech exhibits significant variation from instance to instance for the same speaker and text. An important step in the speaker identification process is to extract sufficient information for good discrimination, and at the same time, to have captured the information in a form and size that is amenable to effective modeling. Speech is usually digitized at a rate of 8kHz or higher, using 8 bits or more per sample, requiring tens of thousands of bytes for a few seconds. It is virtually impossible for a recognition system to take such a huge amount of data as raw inputs, therefore we need to do feature extraction that reduces data dimensions while retaining classification information.

Speech information is primarily conveyed by the short-time spectrum, the spectral information contained in about a 20 ms time period [8, 3]. There are a variety of methods for extract a feature vector from a short-term spectrum of a speech signal with 20 ms. The cepstrum of a segment of speech signals is defined by

$$\text{cepstrum}(frame) = \text{FFT}^{-1}(\log|\text{FFT}(frame)|), \quad (1)$$

where FFT is a fast Fourier transform and $frame$ is a vector of speech signal of 20 ms. Cepstra are better features for speaker recognition since they can better reflect the characteristics of vocal tracts [8, 7].

3. Approaches Employed

This section briefly described our approaches for tackling the speaker recognition problem. These approaches include statistic pattern recognition methods, such as K-nearest neighbor rule, editing and condensing for data reduction, and methods from neuro-fuzzy and soft computing, such as fuzzy c-means clustering and random search. Note that the discipline of soft computing actually encompasses a variety of different methodologies. In this study, we are going to apply only a part of soft computing techniques to

0-7803-3796-4/97/$10.00©1997IEEE

complement the use of the nearest neighbor rule for speaker recognition problems.

3.1. Nearest Neighbor Rule for Classification

The basic idea behind the nearest-neighbor classification rule [4] is that samples which fall close together in the feature space are likely to belong to the same class. To explain the classification process in practice, suppose that we are given a sample data set S_n (also known as the design set) which contains n sample data points:

$$S_n = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\},$$

where $\mathbf{x}_i$ is the feature vector and y_i (which may designate any of the r classes $c_1, c_2, \ldots, c_r$) is the true class of $\mathbf{x}_i$. Now given a new feature vector $\mathbf{x}$, we want to estimate the class of $\mathbf{x}$ based on the sample data set S_n. To apply the nearest neighbor rule (NNR), we first determine the nearest neighbor $\mathbf{x}'$ to $\mathbf{x}$ from S_n, and then assign $\mathbf{x}$ to the class y' of $\mathbf{x}'$. In symbol, the above first nearest neighbor rule (1-NNR) can be expressed as

$$\text{1-NNR}(\mathbf{x}) = y' \text{ if } \delta(\mathbf{x}, \mathbf{x}') = \min_{i=1,\ldots,n} \delta(\mathbf{x}, \mathbf{x}_i),$$

where $\delta(\cdot, \cdot)$ is some metric (for instance, Euclidean distance) of the feature space.

A natural extension to make the classification rule more robust is to use the first several nearest neighbors instead of one. That is, we can find the first k nearest neighbors to $\mathbf{x}$ from S_n, and assign $\mathbf{x}$ to the class that is most heavily represented in the labels of the first k nearest neighbors. Statistic analysis and other variants of the k-nearest neighbor rule (k-NNR) can be found in ref. [4].

3.2. Data Reduction: Editing and Condensing

Assuming that a 256-point frame is adopted to generated a feature vector in the sample data set. Then a 10-second segment of speech signals, sampled at 8kHz, will produce a sample data set of size 624 ($= \frac{10 \times 8000}{256/2} - 1$). Such a large-size design set makes the k-nearest-neighbor rule very demanding in terms of storage space and computation complexity, since each of the sample data must be examined to find the first k nearest neighbors each time a new feature vector is presented to be classified. One way to relieve such constraints is to do data reduction. That is, reducing the number of samples in S_n by retaining a representative subset of the original set. In general, the techniques for data reduction fall into two categories, editing and condensing.

The concept behind editing is to remove inconsistent samples, that is, samples that are close to each other but belong to different categories. The basic step is to randomly select a sample $\mathbf{x}$ and find its nearest neighbor $\mathbf{x}'$, and delete either one of them if they belong to different categories. The editing technique removes inconsistent samples located in class boundary; the remaining samples are usually arranged in homogeneous clusters.

The concept behind condensing is to remove redundant samples, that is, samples that are close to each other and belong to the same category. The basic step is to randomly select a sample $\mathbf{x}$ and find its nearest neighbor $\mathbf{x}'$, and delete either one of them if they belong to the same category. The condensing technique removes deeply embedded samples within the clusters; the remaining samples are usually located at the outer envelopes of the clusters.

Besides editing and condensing, another method for data reduction is vector quantization, which designs a codebook for each speaker from training data using the LBG algorithm, K-means clustering, or fuzzy c-means clustering. For this study, we apply the well-known fuzzy c-means clustering, as described in the next section.

3.3. Fuzzy C-Means Clustering

Fuzzy C-means (FCM) clustering [1] partitions a collection of n vector $\mathbf{x}_i, i = 1, \ldots, n$ into c *fuzzy* groups, and finds a cluster center in each group such that a cost function of dissimilarity measure is minimized. To accommodate the introduction of fuzzy partitioning, the membership matrix U is allowed to have elements with values between 0 and 1. However, imposing normalization stipulates that the summation of degrees of belongingness for a data set always be equal to unity:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \ldots, n. \tag{2}$$

The cost function (or objective function) for FCM is

$$J(U, \mathbf{c}_1, \ldots, \mathbf{c}_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2, \tag{3}$$

where u_{ij} is between 0 and 1; $\mathbf{c}_i$ is the cluster center of fuzzy group i; $d_{ij} = \|\mathbf{c}_i - \mathbf{x}_j\|$ is the Euclidean distance between ith cluster center and jth data point; and $m \in [1, \infty)$ is a weighting exponent. In our simulation, m is set to 2.

The necessary conditions for Equation (3) to reach a minimum can be found by forming a new objective function $\bar{J}$ as follows:

$$\begin{aligned} & \bar{J}(U, \mathbf{c}_1, \ldots, \mathbf{c}_c, \lambda_1, \ldots, \lambda_n) \\ = \; & J(U, \mathbf{c}_1, \ldots, \mathbf{c}_c) + \textstyle\sum_{j=1}^{n} \lambda_j (\sum_{i=1}^{c} u_{ij} - 1) \\ = \; & \textstyle\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 + \sum_{j=1}^{n} \lambda_j (\sum_{i=1}^{c} u_{ij} - 1), \end{aligned} \tag{4}$$

where λ_j, $j = 1$ to n, are the Lagrange multipliers for the n constraints in Equation (2). By differentiating $\bar{J}(U, \mathbf{c}_1, \ldots, \mathbf{c}_c, \lambda_1, \ldots, \lambda_n)$ with respect to all its input arguments, the necessary conditions for Equation (3) to reach its minimum are

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} u_{ij}^m \mathbf{x}_j}{\sum_{j=1}^{n} u_{ij}^m}, \tag{5}$$

and

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}}. \tag{6}$$

The fuzzy C-means algorithm is simply an iterated procedure through the preceding two necessary conditions. In a batch-mode operation, FCM determines the cluster centers $\mathbf{c}_i$ and the membership matrix U using the following steps [1]:

Step 1: Initialize the membership matrix U with random values between 0 and 1 such that the constraints in Equation (2) are satisfied.

Step 2: Calculate c fuzzy cluster centers $\mathbf{c}_i, i = 1, \ldots, c$, using Equation (5).

Step 3: Compute the cost function according to Equation (3). Stop if either it is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

Step 4: Compute a new U using Equation (6). Go to step 2.

The cluster centers can also be first initialized and then the iterative procedure carried out. No guarantee ensures that FCM converges to an optimum solution. The performance depends on the initial cluster centers, thereby allowing us either to use another fast algorithm to determine the initial cluster centers or to run FCM several times, each starting with a different set of initial cluster centers. Bezdek's monograph [2] provides a detailed treatment of fuzzy C-means clustering, including its variants and convergence properties.

3.4. Fuzzy Classifiction System

The purpose of data reduction is to retain a representative subset of the original sample data. The "measure of representativeness" (MOR) of a reduced sample data set may be defined as the recognition rate of the original sample data using the reduced sample data as a design set. Using the MOR scores, we may easily rank the performance of data reduction schemes such as editing, condensing, and VQ-based methods.

Since each point $\mathbf{x}_i$ in the reduced sample set is supposed to be a representative point of the original sample set, we may create a fuzzy if-then rule around the point:

If $\mathbf{x}$ is close to $\mathbf{x}_i$, then the output y is y_i,

where y_i is the output label associated with $\mathbf{x}_i$. The multidimensional antecedent membership function of "close to $\mathbf{x}_i$" in the feature space may be defined as

$$\mu_{\mathbf{x}_i}(\mathbf{x}) = exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{r_i^2} \right),$$

where r_i is the radius of influence of the point $\mathbf{x}_i$. If we have m representative points, we can construct a fuzzy classification system consisting of m rules. Given a new feature $\mathbf{x}$, the fuzzy classification system computes firing strengths from all rules, and assign the output label of the rule with maximal firing strength as the overall output. One may notice that the fuzzy classification system reduces to 1-NNR if r_i is the same for all representative points. On the other hand, if we allow r_i to be different, then we can adjust them to achieve a higher scores of MOR, that is, to make better use of the reduced sample set and thus produce a high recognition rate for the test data set. In this paper, we adjust r_i via random search, as described next.

3.5. Random Search

Random search explores the search space of an objective function sequentially in a seemingly random fashion to find the optimal point that minimizes the objective function. Let $f(\mathbf{x})$ be the objective function to be minimized and $\mathbf{x}$ be the point currently under consideration. The original random search method [6] keeps on evaluating a randomly selected new point and adopting the new point if the objective function is smaller. This is a primitive version in the sense that search directions are purely guided by a random number generator. An improved version that contains a reverse step and a bias term was proposed in ref. [9] and it involves the following steps:

Step 1: Choose a start point $\mathbf{x}$ as the current point. Set initial bias $\mathbf{b}$ equal to a zero vector.

Step 2: Add a bias term $\mathbf{b}$ and a random vector $\mathbf{dx}$ to the current point $\mathbf{x}$ in the input space and evaluate the objective function at the new point at $\mathbf{x} + \mathbf{b} + \mathbf{dx}$.

Step 3: If $f(\mathbf{x} + \mathbf{b} + \mathbf{dx}) < f(\mathbf{x})$, set the current point $\mathbf{x}$ equal to $\mathbf{x}+\mathbf{b}+\mathbf{dx}$ and the bias $\mathbf{b}$ equal to $0.2\mathbf{b}+0.4\mathbf{dx}$; go to step 6. Otherwise, go to the next step.

Step 4: If $f(\mathbf{x} + \mathbf{b} - \mathbf{dx}) < f(\mathbf{x})$, set the current point $\mathbf{x}$ equal to $\mathbf{x} + \mathbf{b} - \mathbf{dx}$ and the bias $\mathbf{b}$ equal to $\mathbf{b} - 0.4\mathbf{dx}$; go to step 6. Otherwise, go to the next step.

Step 5: Set the bias equal to 0.5b and go to step 6.

Step 6: Stop if the maximum number of function evaluations is reached. Otherwise go back to step 2 to find a new point.

Usually the initial bias is set to a zero vector. Each component of the random vector **dx** should be a random variable that has a zero mean and a variance proportional to the range of the corresponding parameter; this allows the method to apply the same degree of exploration for each dimension of the parameter space. A detailed coverage of random search with examples can be found in ref. [5].

4. Experimental Settings and Results

4.1. Data Acquisition and Feature Extraction

The speaker recognition experiments were conducted on a data set of 5 male speakers; each speaker contributed three sentences to the data set and each sentence lasts for about 2 to 3 seconds. All the data acquisition was performed using the recording program under Windows 95; the sampling frequency is 8kHz with a 8-bit A-D. The underlying platform is a Pentium-166 PC, with a dSPACE 1102 controller board that can eventually take speech signals from a speaker, do FFT (fast Fourier transform) and others to extract features, feed the features to a trained classifier, and return the identity of the speaker in a real-time fashion.

The feature set used for encoding the speech signal was a form of cepstral coefficients computed as follows:

1. Partition the speech signals into disjoint half-frames of 128 points. Neighboring half-frames are joined to form a complete frame of 256 points. (In other words, each half-frame is presented in two complete frames, and the intersection of two consecutive frames is a 128-point half-frame.) Since the sampling frequency is 8 kHz, each frame lasts $256/8 = 32$ ms.
2. Window each frame using a 256-point Hamming window to lessen distortion.
3. Compute the cepstral coefficients of each frame using Equation (1).
4. The first 14 coefficients of the cepstrum is taken as the feature vector of the frame.

In our simulation, we used the first sentences of the speakers as the training set (or design set), the second and third sentences as the test set. The above process can thus extract 578 training entries and 1063 test entries; each entry contains a feature vector of 14 elements and a output label that may assume the value of 1, 2, or 3. The following table lists the constituents of the training and test sets.

	Speaker 1	Speaker 2	Speaker 3	Total
Training set	148	280	150	578
Test set	256	457	350	1063

4.2. Experimental Results

4.2.1. k-NNR without data reduction

Without using any data reduction techniques, the performance of k-NNR (k nearest-neighbor rule), with $k = 4$, is shown as the following confusion probability matrix C, where c_{ij} denotes the probability of a feature vector of class i being classified as class j:

	Speaker 1	Speaker 2	Speaker 3
Speaker 1	83.5%	5.8%	10.5%
Speaker 2	17.9%	73.9%	8.0%
Speaker 3	13.1%	10.5%	76.2%
Overall recognition rate = 77.9%			

At a first glance, it may seem that the recognition rates in the preceding table are on the low-end side. However, those percentages are based on single-frame recognition and higher recognition rates can be achieved if multiple consecutive frames are used in a voting mechanism to find a overall computed class. A single frame corresponds to a speech signal of 32-ms duration; an m-frame ensemble corresponds to a speech signal of $((m+1)*16)$ ms. Figure 2 demonstrates how the recognition rates increase with the input lengths (or equivalently, numbers of consecutive frames).

The upper plot in Figure demonstrate the recognition rates for test data from different speakers, as the input length varies from 32ms to about 2 sec. Apparently speaker 1 is the easiest one to identified, while speaker 2 is the most difficult. The lower plot in Figure shows the overall recognition rates as a function of the input length. A recognition rate of 100% can be achieve when the input length is longer than about 0.6 sec (or equivalently, about 40 frames).

The performance in this experiment serve as a benchmark for other variants of data reduction methods and/or discriminative functions described next.

4.2.2. k-NNR with data editing and condensing

As described earlier, the editing technique removes samples in the boundary areas, while the condensing technique removed samples deeply embedded in a same-class cluster. Thus it is a common practice to perform editing first and then condensing. In our simulation, the editing technique

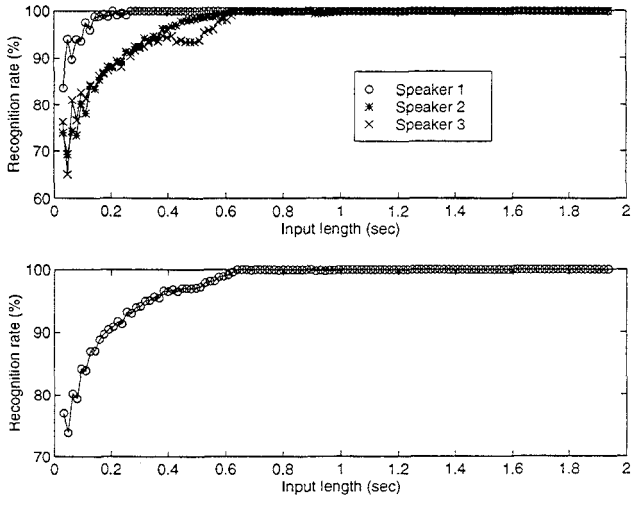

Figure 1. *Recognition rates of 4-NNR using the entire sample data. (Upper: recognition rates for individual speakers; lower: overall recognition rates.)*

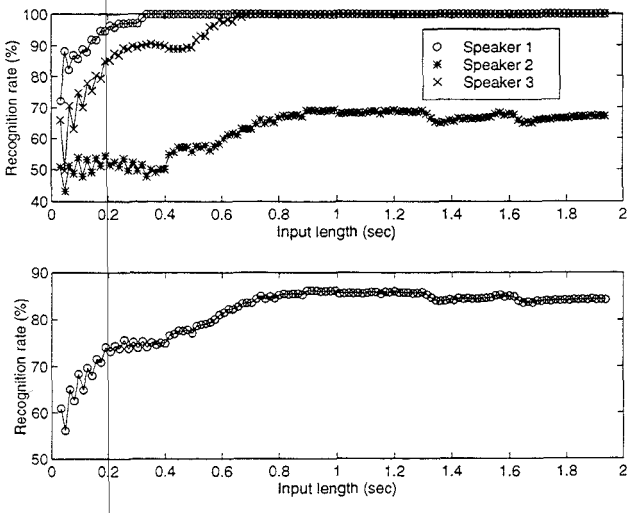

Figure 2. *Recognition rates of 4-NNR using a reduced sample data set obtained via editing and condensing. (Upper: recognition rates for individual speakers; lower: overall recognition rates.)*

reduced the training set size from 578 to 496, and the condensing technique further reduced it to 75. The resulting confusion matrix is shown next.

	Speaker 1	Speaker 2	Speaker 3
Speaker 1	72.2%	10.9%	16.7%
Speaker 2	41.3%	50.7%	7.8%
Speaker 3	13.4%	20.5%	66.0%
Overall recognition rate = 63.0%			

The recognition rate curves are in Figure 2. The recognition rates in this case are generally lower, but the training set size is much smaller (75 compared to 578) and the computation time of k-NNR is significantly reduced. Obviously the reduced sample set after editing and condensing is not really a good representative subset. In particular, the recognition rate for speaker 2 is very low, which makes the overall recognition rate curve below 100% even when the input length is 2 sec.

4.2.3. k-NNR with data reduction via FCM

In contrast, we applied the FCM (fuzzy c-means) clustering [2] to each of the three classes of the original 578 training samples. The cluster centers found by FCM is then used as the design set for k-NNR with $k = 4$. To make a fair comparison, the total number of cluster centers is equal to 75, and the number of clusters in each class is proportional to the number of training data in the class. The following table is the confusion probability matrix.

	Speaker 1	Speaker 2	Speaker 3
Speaker 1	77.7%	13.2%	8.9%
Speaker 2	20.5%	71.7%	7.6%
Speaker 3	13.7%	14.8%	71.4%
Overall recognition rate = 73.6%			

In general, the recognition rates are higher than the previous case, which indicates that the reduced sample set by FCM is more consistent than the one by editing and condensing. Figure 3 is the corresponding recognition rate curves. The overall recognition curve reaches 100% when the input length is around 1.2 sec.

4.2.4. Fuzzy classification system

In this experiment, we applied FCM to find a reduced sample set of size 75, and then create a 75-rule fuzzy classification system in which the width of each rule is adjusted via random search to maintain the consistency between the original and reduced sample sets. The resulting confusion probability matrix is

	Speaker 1	Speaker 2	Speaker 3
Speaker 1	57.8%	30.0%	14.0%
Speaker 2	8.7%	82.5%	7.6%
Speaker 3	9.7%	9.4%	82.8%
Overall recognition rate = 75.4%			

Figure 4 is the corresponding recognition rate curves. Because of the use of random search to adjust r_i of each

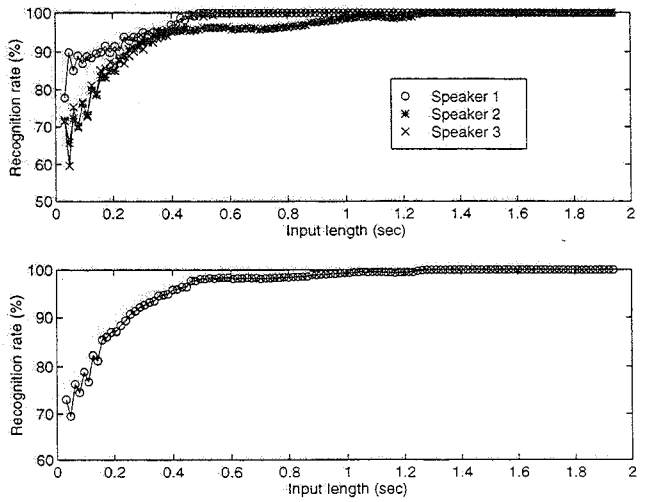

Figure 3. *Recognition rates of 4-NNR using a reduced sample data set obtained via FCM.*

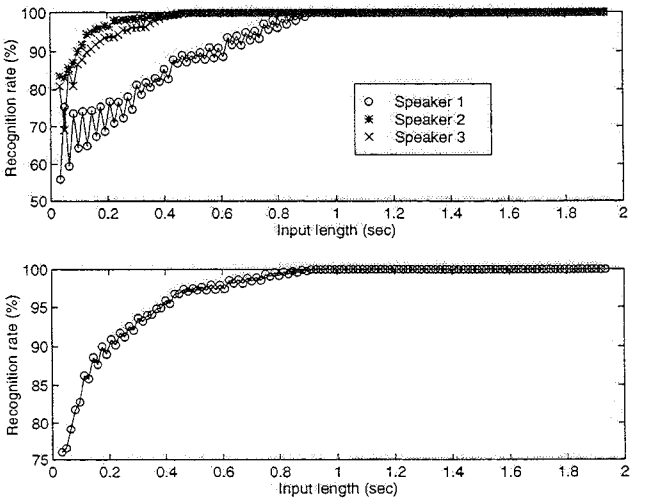

Figure 4. *Recognition rates of the fuzzy classification system which applies random search for adjusting r_i of each rule.*

rule, the recognition rates are generally higher and the overall recognition rate reaches 100% when the input length is around 0.9 sec, which is close to the benchmark case (0.6 sec.), and better than those by FCM alone (1.2 sec.) and editing plus condensing (more than 2 sec.).

5. Concluding Remarks and Future Work

In this paper, we have successfully applied several soft-computing techniques to enhance the performance of the nearest neighbor classification rule for speaker recognition. It is demonstrated that, with enough computing power, the fuzzy c-means clustering can effectively find representative sample points, which can then be used as a design set without degrading the recognition rates too much. Further improvements can be obtained if we apply random search to find the near-optimal radius of each sample point.

This is only a starting point of using soft computing techniques for digital signal processing and pattern recognition. Other potential directions include:

- Use gradient-free optimization to select the best frame size for speaker recognition. (In this paper, the frame size is fixed at 256 points.)
- Use gradient-free optimization to select the most relevant features for further classification. (In this paper, the number of features is the first 14 cepstral coefficients.)
- Assign different radius of influence to each feature and apply gradient-free optimization to find their best values. (In this papers, the radius of influence is the same for each features for a given representative point.)

Moreover, the experiences we gained from the experiments pave he avenue to a set of more difficult biometric identification problems, including recognition of faces, fingerprints, palm prints, retinal blood-vessel patterns, and so on.

References

[1] J. C. Bezdek. *Fuzzy Mathematics in Pattern Classification*. PhD thesis, Applied Math. Center, Cornell University, Ithaca, 1973.

[2] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum Press, New York, 1981.

[3] J. R. Deller, J. G. Proakis, and J. H. L. Hansen. *Discrete-time processing of speech signals*. Macmillan, New York, 1993.

[4] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, Upper Saddle River, NJ, 1982.

[5] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. MATLAB Curriculum Series. Prentice Hall, Upper Saddle River, NJ, 1997.

[6] J. Matyas. Random optimization. *Automation and Remote Control*, 26:246–253, 1965.

[7] A. M. Noll. Cepstrum pitch determination. *Journal of the Acoustical Society of America*, 41:293–309, 1967.

[8] L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition*. Prentice Hall, Upper Saddle River, NJ, 1993.

[9] F. J. Solis and J. B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.