Classification of Audio Radar Signals Using Radial Basis Function Neural Networks

Trent McConaghy, *Member, IEEE*, Henry Leung, *Member, IEEE*, Éloi Bossé, and Vinay Varadan

***Abstract*—Radial basis function (RBF) neural networks are used to classify real-life audio radar signals that are collected by a ground surveillance radar mounted on a tank. Currently, a human operator is required to operate the radar system to discern among signals bouncing off tanks, vehicles, planes, and so on. The objective of this project is to investigate the possibility of using a neural network to perform this target recognition task, with the aim of reducing the number of personnel required in a tank. Different signal classification methods in the neural net literature are considered. The first method employs a linear autoregressive (AR) model to extract linear features of the audio data, and then perform classification on these features, i.e, the AR coefficients. AR coefficient estimations based on least squares and higher order statistics are considered in this study. The second approach uses nonlinear predictors to model the audio data and then classifies the signals according to the prediction errors. The real-life audio radar data set used here was collected by an AN/PPS-15 ground surveillance radar and consists of 13 different target classes, which include men marching, a man walking, airplanes, a man crawling, and boats, etc. It is found that each classification method has some classes which are difficult to classify. Overall, the AR feature extraction approach is most effective and has a correct classification rate of 88% for the training data and 67% for data not used for training.**

***Index Terms*—Audio signal, classification, neural net, radar, radial basis function.**

I. INTRODUCTION

THIS PAPER reports the result of a project supported by the Department of National Defence of Canada to investigate the possibility of replacing a trained human operator by an artificial neural network in performing radar target recognition. Radar is still an important sensor in current defence systems for sensor surveillance, since radar is the only sensor which can effectively detect targets at long distances, in the dark and almost any weather conditions [1]. Currently, the identification function in a radar system is usually carried out by human operators who have had special training. However, the amount of incoming data is already far more than humans can handle, and the amount of available manpower is shrinking. As well, the reduction of crew size in operating a surveillance system with no reduction in the surveillance performance is beneficial. Development of an automated system to carry out radar identification function is therefore highly desirable [2]–[4].

In this project, we consider radar signals collected by battlefield radar systems that are installed in a tank. These signals are at audio frequencies, and a trained human operator is usually required to stay inside the tank to recognize these audio radar signals. The data set that we use was collected by an AN/PPS-15 ground surveillance radar. In this type of radar, a microwave continuous wave signal of constant frequency is transmitted and is used to form a local oscillator for the received signal. The output signal can then be reduced to audio frequencies and represents the Doppler shift incurred because of target motion along the radar line of sight [5]. There are totally 13 classes of signals from various moving targets in this experiment, including trucks, tanks, men marching, a man walking, a man crawling, boats, small airplanes, and birds.

Neural networks are investigated here for classifying the audio radar signals automatically due to its efficiency in processing audio signals [6], [7]. In particular, the radial basis function (RBF) neural network is used as the neural net classifier in this study [8]. The main reason is that a linear adaptive algorithm can be used in training the coefficients of the network. This makes the RBF net implementable for real-time application. For other neural networks such as recurrent neural networks [9]–[11], the training process may not be carried out in a real-time on-line fashion using current technologies [12]. Because audio signals are usually nonstationary, this on-line learning ability is preferred to allow the classifier being adaptive to the environment. In addition, an RBF is a universal approximator which has the capability of approximating a decision boundary of any shape.

Based on our survey, we find three popular approaches for time series classification using neural networks. The main difference is the feature extraction process. The first approach applies a neural network classifier to the raw data directly without any separate feature extraction procedure. The input is basically the delayed values of the time series $\{x(t), t = 1, 2, 3, \ldots\}$, i.e., $\{(x(t), x(t+1), \ldots, x(t+(d-1))), t = 1, 2, \ldots\}$ for a network with d input units. However, the raw data approach usually has a poor performance and we have the same observation for this audio radar classification problem. The detailed analysis of this approach is therefore omitted due to space limit. The second approach uses a model to represent the data, and applies a neural network classifier on the model parameters. Fig. 1 illustrates this approach. Here,

Manuscript received October 1, 2001; revised July 27, 2003.

T. McConaghy is with the Analog Design Automation, Inc., Ottawa, Ontario, ON K2P 2C2 Canada (e-mail: trent@analogsynthesis.com).

H. Leung is with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, Alberta, AB T2N 1N4 Canada (e-mail: leungh@enel.ucalgary.ca).

E. Bossé is with the Decision Support Technology Section, Defence Research and Development Canada, Valcartier, Quebec, QC G3J 1X5 Canada (e-mail: eloi.bosse@drdc-rddc.gc.ca).

V. Varadan is with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: varadan@ee.columbia.edu).

Digital Object Identifier 10.1109/TIM.2003.820450

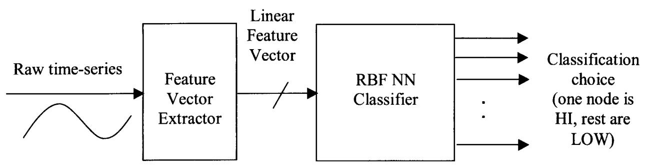

Fig. 1. Feature vector approach to classification.

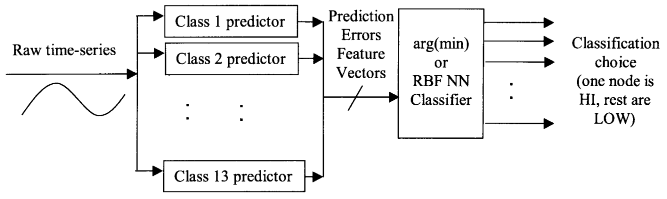
Raw time-series
Class 1 predictor
Class 2 predictor
Class 13 predictor
Prediction Errors Feature Vectors
arg(min) or RBF NN Classifier
Classification choice (one node is HI, rest are LOW)

Fig. 2. Prediction approach to classification.

we use the popular autoregressive model (AR) to extract the features of these audio signals [13], [14]. The AR parameters are estimated in two ways: i) using the least square (LS) and ii) using the cumulant method [13], [15]. The third approach models the audio radar time series using a predictor, the predictor could be either linear [16] or nonlinear [17], [18]. Instead of performing classification on the model parameters, this third approach performs classification on the prediction errors [19]. As shown in Fig. 2, this predictive classification approach requires N_c predictors to classify N_c different classes of signals. A signal that has a minimum prediction error from the ith predictor is expected to belong to class i if the predictor is a good model for the data. It is noted that these two approaches are considered here since they cover the two main categories of signal representations: linear and nonlinear. While the AR feature approach is still the most standard method for audio and speech signal modeling, the nonlinear method represents another major approach for processing audio signals [17], [18].

Our classifier's "training and testing" procedure is similar to the way human operators learn to classify this audio radar data. In a classroom setting, humans "train" their ears on training data so that they recognize certain sounds as belonging to certain classes. Afterward, the humans are tested on separate recordings. The performance of these classifiers is then evaluated based on both training and testing data.

Section II of this paper describes the radar data in more detail. Section III presents the classification methods based on AR coefficients and prediction. Section IV describes the RBF classifier along with its associated training techniques (to find optimal connection weights and the number of parameters). Section V presents the performance and analyzes of the neural network approaches on classifying the audio radar signals. Concluding remarks are given in Section VI.

II. Description of the Radar Data

The audio radar data used here are collected by an AN/PPS-15 ground surveillance radar. The parameters of this radar system are given as follows.

Radar power	100 mW.
Antenna gain	29.2 dB.
Frequency	10.3 GHz.
Bandwidth	1 kHz.
Antenna area	0.06 m^2.
Signal-to-noise ratio	10 dB.
Target radar cross section	10.0 dBsm.

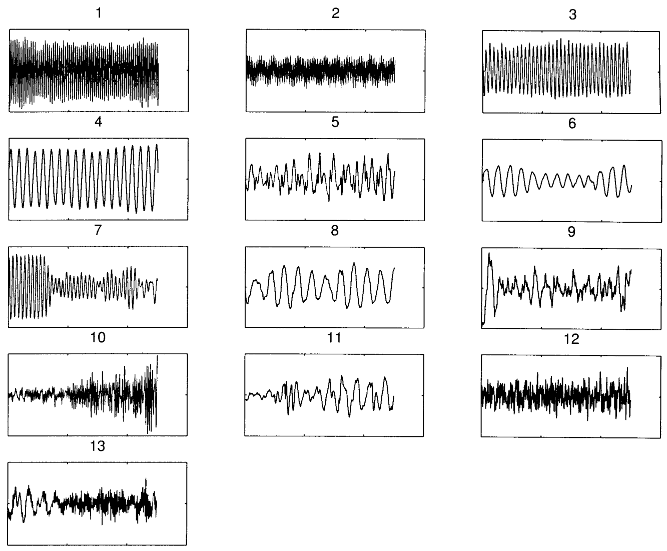

Fig. 3. Typical waveforms of the 13 classes of radar signals. Each segment contains 500 points.

There are $N_c = 13$ classes of targets for classification. The time-series samples of these signals are shown in Fig. 3 for illustration. The details of these target signals are described as follows.

Class number	Description of signal
1	vehicle at 45 mph, moving away from radar;
2	vehicle at 20 mph, moving toward radar;
3	vehicle at 10 mph, moving away from radar;
4	vehicle at 5 mph, moving away from radar;
5	tracked vehicle moving at right angles, stopping often;
6	several men marching;
7	man running toward radar;
8	man walking toward radar;
9	man crawling toward radar;
10	boat with outboard motor;
11	heavy clutter;
12	propeller-type plane;
13	large birds, coming and going; noise.

These 13 classes of targets are of particular interest to the military people who provide the data. Since the radar uses the Doppler to obtain the audio signals, slight variation in target characteristic within a class such as two men walking instead of one will not cause serious changes in the classification performance. Each of these classes had an associated audio recording, captured to disk at a sampling frequency of 8 kHz. Upon examination of the spectra of each of the signals, we found that we could resample all the data to 4 kHz without loss of information. Each class had 12 000 data points (i.e., 3 s). The first 9000 points were used for training data, and the remaining 3000 points were designated for testing. We use 9000 training points as the training set as that would mean using close to 66% of the available data for training and would still leave around 33% of the recorded data for testing. Increasing the training data size any further would mean a reduction in the available testing data sample, thus reducing our ability to generalize the performance of the classifiers used. From the training data, 300 training vectors from each class were created according to the various approaches. There were 100 vectors from each class for testing.

III. Extracting Features of the Audio Signals

A. Autoregressive (AR) Coefficients

Two well-known methods of AR estimation techniques are considered: i) the least square (LS) technique, which is optimal for signals with Gaussian distribution, and ii) the cumulant technique [13], which works better for non-Gaussian signals. For each AR technique, the model orders calculated ranged from size 2 to 15.

We calculated the AR coefficients as follows. If the past M points of a discrete time series $x(n)$ are represented by $x(n-1), x(n-2), \ldots, x(n-M)$ and have a mean of zero, and if the AR coefficients are $a_1, a_2, \ldots, a_N$, then the next value $x(n)$ can be approximated by

$$x(n) = \sum_{i=1}^{M} a_i x(n-i). \quad (1)$$

If the data sequence has N points, we can express the system in the following matrix:

$$\begin{pmatrix} x(M+1) \\ x(M+2) \\ \vdots \\ x(N) \end{pmatrix} = \begin{pmatrix} x(M) & x(M-1) & \cdots & x(1) \\ x(M+1) & x(M) & \cdots & x(2) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & \cdots & x(N-M) \end{pmatrix} \times \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{pmatrix}. \quad (2)$$

We assign the names $\mathbf{y}$, $\mathbf{X}$, and $\mathbf{a}$ to the above matrices, such that $\mathbf{y} = \mathbf{Xa}$. The LS-AR coefficients may be obtained by solving the normal equation directly

$$\mathbf{a} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}. \quad (3)$$

This estimate of the AR coefficients for a normal model has been shown to have zero estimation bias as well as minimum estimation variance. Thus, in the case of the normal model assumption, this estimator is optimal in not only a least squares (LS) sense but also in a minimum mean square error (MSE) sense [20].

As shown in [13], the cumulant-based AR coefficients can be obtained by solving the following equations [21]:

$$\sum_{i=0}^{M} a_i C_{2x}(j-i) = 0, \quad j > 0$$
$$\sum_{i=0}^{M} a_i C_{3x}(j-i, p) = 0, \quad j > 0 \quad (4)$$

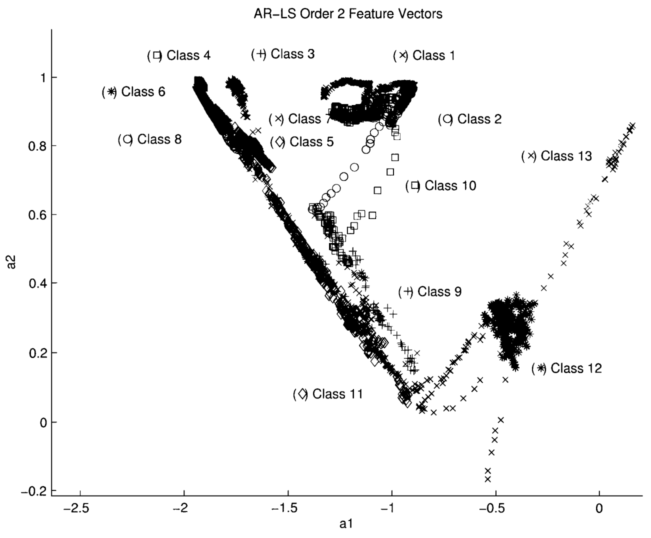

Fig. 4. Display of the AR-LS feature vectors. The order of the AR model is 2 and each class contains 300 vectors.

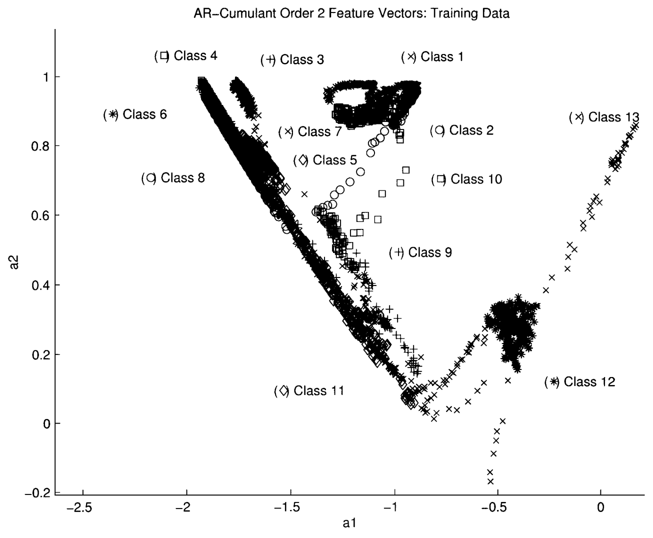

Fig. 5. Display of the AR-cumulant feature vectors. The model order is 2 and each class contains 300 points.

where $a_0 = 1$. The second-order cumulant, $C_{2x}(i)$, and third-order cumulant, $C_{3x}(i,j)$, are defined as

$$C_{2x}(k) = E\left[x^*(n)x(n+k)\right]$$
$$C_{3x}(k,l) = E\left[x^*(n)x(n+k)x(n+l)\right]. \quad (5)$$

Note: $E[\cdot]$ is the expectation operator. The LS solution is given by the first equation in (4) with $j = 1, \ldots, M$ (the normal equation) and the third-order cumulant solution can be obtained by (4) for $p = -M, \ldots, 0$ and $j = 1, \ldots, M$.

Figs. 4 and 5 show the feature vectors extracted using a second order AR model. A sample size of 300 was used in the calculation of AR coefficients. We can see that class 1, a vehicle at 45 mph, overlaps somewhat with class 2, a slower vehicle. Examining further, we see that many classes overlap each other. This sort of overlap is apt to confuse the network when training. However, there are classes that should be easy for the automatic classifier to distinguish from the rest. For example, we see that class 12, a propeller-type plane, is well separable from other classes. Class 13 is composed of many birds coming and going with a lot of background noise. We see that the feature vectors for this class move about the two-dimensional (2-D) space quite a bit. Accordingly, we expect the classifier to confuse this class often.

Figs. 4 and 5 differ by the method of calculation for linear models. Fig. 4 displays feature vectors that are calculated by the LS method. Fig. 5 has feature vectors that are calculated with a cumulant-based approach. We see that there is very little visible difference in the feature vector diagrams; from this we expect that there may be little difference in the final classification results based on these two feature vector extraction approaches.

We have seen that classes overlap when represented as 2-D linear models. We hope that higher model orders will increase the separability of these classes. Hence, we will be trying a variety of model orders, ranging from dimension 2 to dimension 15. In addition, we have seen that some classes should be easier to correctly classify than others. Accordingly, the neural network training will require more effort to learn to divide "trouble" classes. We keep this in mind when choosing an algorithm to train the neural network.

B. Prediction Error

The basic idea behind classification based on time series prediction is that if we can accurately predict future values of a signal using a predictive model from a known class, then that signal is likely to be from that class as well. That is, in a set of N_c predictors (for classes 1, 2, ..., and N_c respectively), if predictor i has the lowest prediction error on a given input signal, then it is very possible that the input signal is from class i. This predictive approach for neural network classification has recently been found to be quite effective in signal classification, especially for signals with strong nonlinearities [19].

The time series prediction approach can be broken into three segments: a) finding optimal linear or nonlinear predictors for each class, b) creating feature vectors from an input class based on the prediction error of each predictor, and c) classifying the feature vectors.

To find the optimal linear predictors, we broke the training data into two halves. For each possible model order $M = \{2, 3, \ldots, 24\}$, we trained a predictor on the first half of the data, then measured prediction error on the second. Linear coefficients were calculated using the LS method. From the prediction error, we calculated the final prediction error (fpe) value as follows:

$$fpe(\nu, N, M) = \frac{\nu(N + M - 1)}{N - M - 1} \quad (6)$$

where v is the estimated variance of the linear predictor's errors, N is the length of the sequence from which the AR model was estimated (in our case 300), and M was the model order that we are trying to find, that gives us the lowest fpe. Once we found the optimal M, we calculated the best linear representation over all the training data. We have used this definition of fpe to determine the model order as it does not assume any a priori distribution of the underlying data in contrast to the minimal description length (MDL) and Akaike information criterion (AIC) [20], [21] where one needs to assume a probabilistic distribution of the data. Thus, our approach to finding the optimal M is based on the mean square prediction error performance of the model, which is more general than the MDL or AIC criteria.

To find the best nonlinear predictor for each class where an RBF is used for this purpose here, we tried different numbers of RBF centers, from {10, 20, ..., 100}. We set the embedding dimension to 24. The number of RBF centers for each class turned out to be quite different for different targets, and it depends on the complexity of the signal. The predictors were given the same training data as for the linear approach; the optimal predictor was chosen as the one that had lowest mean square prediction error on the cross validation training data. Keeping the same number of RBF centers, it was trained across all the training data. Here too, we used the minimum MSE on the cross validation training data as that is a very general criterion for prediction performance without making any assumption of the distribution of the underlying data.

Once we found the 13 optimal predictors (one for each class), we used them to create feature vectors of length 13. Given an input signal, each of the predictors would try to predict the signal's next value, and there would be an associated prediction error. However, because the signals had noise, the examination of just one prediction error is not enough to classify with. Predictors trained to classify a signal may fare poorly, or other predictors may get a lucky prediction. This problem becomes alleviated if we average prediction error over more than one sample. Since we had 9000 samples, a maximum predictor order of 24 and wanted to use 300 training vectors, we calculated that we could use floor $(9000 - 24)/300 = 29$ prediction errors from each predictor to create a training vector. So, from each predictor i, we squared 29 prediction errors, then took the average, to create the entry for location i in the current vector p being created.

Due to space limit, we only show the feature vectors created by the nonlinear predictors for one class of signals in Fig. 6. Note that the brightness is proportional to magnitude of squared error. Since the signal was taken from class 1, it can be seen that the prediction error of the class one predictor is always lower compared to the other predictors. It can, however, be seen that the prediction error by other class predictors such as class 10 and class 2 predictors is also low. These sorts of predictors could cause confusion in the decision module. We should, however, stress that this problem can be alleviated through the use of a neural network classifier trained to handle such ill-posed problems.

Once we created the prediction-based feature vectors for both training and testing data, we could apply the classifier. The simplest classification decision scheme was to simply choose the class k corresponding to the predictor that had the lowest prediction error, i.e., the k that produces the minimal. This classification scheme has the merit that a new classifier does not need to be trained with the addition of new classes. This scheme is also the most scalable to larger numbers of classes. We also considered the possibility that predictors of the correct class may not always have the lowest prediction error. Certain patterns may exist in the feature vectors that distinguish classes from each

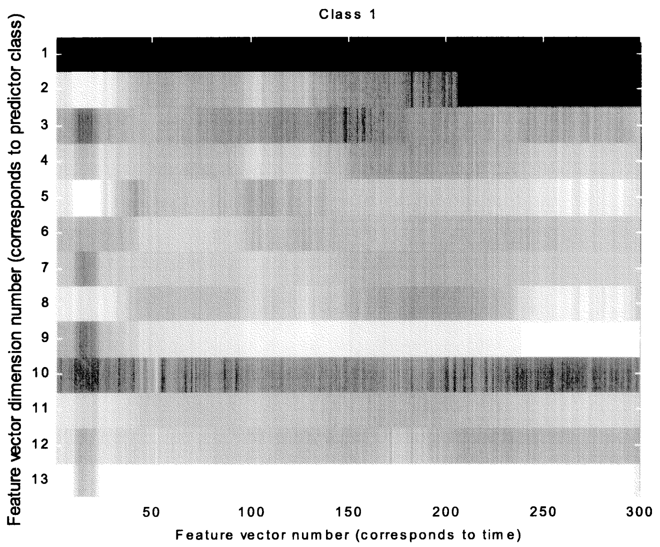

Fig. 6. Feature vectors generated by the nonlinear predictor in predicting the class 1 signal.

other in a more complicated way than simply choosing the minimum. We wanted to create a classifier that is trained to classify based on more complete patterns. So, we trained and tested a neural network classifier in a manner identical to the AR feature vector extraction approach, except we used the prediction error feature vector instead of the AR feature vectors.

IV. Classification Using a RBF Neural Network

This section describes how we used an RBF network to classify the audio radar signals. The RBF NN used here has an input layer, a hidden layer consisting of Gaussian node functions, an output layer, and a set of weights, $\boldsymbol{W}$, to connect the hidden layer and the output layer. We denote $\mathbf{x}$ to be the input vector to the network, where $\mathbf{x} = (x_1, x_2, \ldots, x_D)^T$, and D is the embedding dimension. We call $\boldsymbol{o}$ the ANN output vector, where $\mathbf{o} = (o_1, o_2, \ldots, o_{Nc})^T$ is the number of output nodes. We have $\boldsymbol{P}$ training patterns. The RBF classification problem is to approximate the mapping from the set of inputs, $\mathbf{X} = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(P)\}$, to the set of outputs, $\mathbf{O} = \{\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(P)\}$.

For an input vector $\mathbf{x}(t)$, the output of the jth output node produced by an RBF is given by

$$o_j(t) = \sum_{i=1}^{m_{tot}} w_{ij}\phi_i(t) = \sum_{i=1}^{m_{tot}} w_{ij} e^{-\frac{\|\mathbf{x}(t) - \mathbf{c}_i\|}{2\sigma_i}} \tag{7}$$

where $\boldsymbol{c}_i$ is the center of the ith hidden node, σ_i is the width of the ith center, and m_{tot} is the total number of hidden nodes. Using vector notation, let

$$\boldsymbol{\varphi} = (\phi_1(t), \phi_2(t), \ldots, \phi_{m_{tot}}(t)) \tag{8}$$

and

$$\mathbf{w_j} = (w_{1j}, w_{2j}, \ldots, w_{m_{tot}j}) \tag{9}$$

the RBF output can be written as

$$o_j = \mathbf{w_j}\boldsymbol{\varphi}^T(t). \tag{10}$$

The RBF classifier contains four sets of parameters that have to be learned from the examples. They are the centers, $\mathbf{c}_i(t)$, number of centers m_{tot}, variances σ_i, and weights w_{ij}. We denote all the RBF NN's centers by $\boldsymbol{C}_{whole}$. Each class c has an associated set of RBF centers, $\mathbf{C_c} \in \mathbf{C_{whole}}$. In our implementation of RBF classifier, classes do not share centers. Each of these sets of centers is trained with a separate K-mean clustering run. In each K-mean run (corresponding to a different class), only the training vectors for that class would be used for clustering.

The K-mean approach for one set of centers $\boldsymbol{C_c}$ is now described. In this approach, the centers are initially defined as the first m_c training inputs that correspond to class c; that is, $\mathbf{C}_c(i = 0) = \{\mathbf{x}(c_1), \mathbf{x}(c_2), \ldots, \mathbf{x}(c_{m_c})\}$, where $\mathbf{X_c} = \{\mathbf{x}(c_1), \mathbf{x}(c_2), \ldots, \mathbf{x}(c_{P_c})\}$ and $\boldsymbol{P_c}$ is the number of training patterns for class c. At each iteration i following, a new training input, $\mathbf{x}(i)$, is presented. The distances for each of the centers are denoted by $\rho_j(i)$, $j = 1, 2, \ldots, m_c$. That is

$$\rho_j(i) = \|\mathbf{x}(i) - \mathbf{c_j}(i-1)\|. \tag{11}$$

The kth center is then updated according to these distances via the following:

$$\mathbf{C_k}(i) = \mathbf{C_k}(i-1) + \alpha\,|\rho_k(i)| \tag{12}$$

where "k" is chosen as the k that minimizes $\rho_j(i)$, that is

$$k = \arg(\min(\rho_j(i))) \tag{13}$$

and α is the learning rate. Here we set $\alpha = 0.1$.

The width associated with the kth center is adjusted according to the N_c centers adjacent to $\boldsymbol{C_k}$. In our implementation, we set $N_a = 2$

$$\sigma_k(i) = \sqrt{\frac{1}{N_a}\sum_{j=1}^{N_a}\|\mathbf{C_k}(i) - \mathbf{C_j}(i)\|^2}. \tag{14}$$

With the estimated centers and variances, the weights of the RBF classifier can be trained using the linear recursive least squares (RLS) algorithm. The RLS is employed here since it has been shown to have a much faster convergence rate than the gradient search type algorithm such as the least mean squares (LMS) algorithm [22]

$$\begin{aligned}\mathbf{k}(i) &= \frac{\mathbf{P}(i-1)\boldsymbol{\varphi}^T(i)}{\lambda + \boldsymbol{\varphi}(i)\mathbf{P}(i-1)\boldsymbol{\varphi}^T(i)}\\ \mathbf{w_j}(i) &= \mathbf{w_j}(i-1) + \mathbf{k}(i)\left[\mathbf{d_j}(i) - \mathbf{w_j}(i-1)\boldsymbol{\varphi}^T(i)\right]\\ \mathbf{P}(i) &= \frac{1}{\lambda}\left[\mathbf{P}(i-1) - \mathbf{k}(i)\boldsymbol{\varphi}(i)\mathbf{P}(i-1)\right]\end{aligned} \tag{15}$$

where λ is a real number between 0 and 1, $\mathbf{P}(0) = a^{-1}\mathbf{I}$ and a is a small positive number, and $\hat{\mathbf{w}}_j(0) = \mathbf{0}$.

There are several reasons for using an RBF net in our audio radar signal classification problem. First, many neural networks require nonlinear optimization for training. Although many researches have been conducted to improve the efficiency of nonlinear optimization, their computational complexity is still relatively high. For the real-time radar application considered in this study, a fast learning neural network is a necessity.

Since the weights of an RBF can be obtained by using a linear adaptive filtering algorithm, it is an appropriate candidate to process the radar audio signals. This adaptive learning ability is essential for real-time radar signal processing and for tracking the nonstationarity of these audio signals.

The second reason for employing a RBF classifier is that the internal representation of training data of an RBF is intuitive. Each RBF center approximates a cluster of training data vectors that are close to each other in Euclidean space. When a vector is input to the RBF NN, the centers near to that vector become strongly activated, in turn activating certain output nodes.

Because of this last property of RBF, we propose controlling the number of centers assigned to each class in our classifier. When we have "trouble" classes, we will assign more centers to the regions of input space that the trouble classes occupy. The network then refines the boundaries between trouble classes, thereby making the RBF NN a more accurate classifier. This idea can also be thought of as "growing new subspaces" in the network's internal representation [23].

Let m_c be the number of centers assigned to class c; then $\mathbf{m} = \{m_1, m_2, \ldots, m_{N_c}\}$ stores the number of centers assigned to each of the classes. We used the following algorithm to automatically find the optimal number of centers needed for each class, for each classification approach, and each embedding dimension of inputs.

1) For each c, $c = 1, 2, \ldots, N_c$, assign an initial number of centers to each class: $\boldsymbol{m}_c = \boldsymbol{m}_{init}$.
2) Train the RBF NN using the current set of centers assigned to each class, $\boldsymbol{m}$, to get cross validation misclassification error, e_c for each class c, $\mathbf{e} = \{e_1, e_2, \ldots, e_{N_c}\}$.
3) Let $e_m = mean(\mathbf{e})$. If $e_m < e_{m,target}$ or e_m has not decreased by more than 0.2% over the last I iterations, goto 5.
4) Add m_{inc} centers to the N_{ec} classes with the highest error, to get a new $\boldsymbol{m}$. Goto 2.
5) The RBF NN to use is the one with the current $\boldsymbol{m}$.

Step 1) initializes the algorithm. Step 2) requires the training of an RBF NN in the usual way, except that clustering occurs N_c times. Step 3) is a convergence test and a stagnation test. If the NN's classification error rate is lower than the pre-specified target rate, convergence will halt. Also, if the error rate is no longer dropping, convergence will halt. Step 4) guides the NN's complexity growth, based on the "trouble" classes. Reaching step 5) indicates completion. It is the NN returned by this step that is used in the test data.

In our work, we set the parameters as follows. For all classification approaches, we set $e_{m,target}$, the target misclassification rate, to 5% and I to 5. For the AR feature approach, since the size of the AR feature vector is relatively smaller, we set $m_{init} = 1$, $m_{inc} = 1$, and $N_{ec} = 5$. For the prediction approach, each neural network was a model of just one class. Therefore, for this approach, there was no need to assign centers corresponding to each class. To find the optimal predictor, we used the above approach, but $\boldsymbol{m}$ only contained one m, to represent the total number of centers for the RBF NN. Also, prediction error replaced classification error. We set $m_{init} = 5$, $m_{inc} = 2$, and $N_{ec} = 1$. This basically means that we started testing predictors with $m = m_{init}$, and kept adding m_{inc} new centers and re-testing until stagnation occurred.

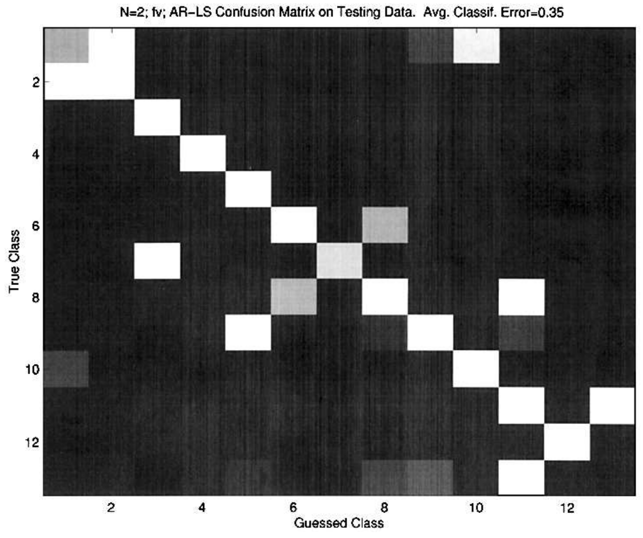

Fig. 7. Confusion matrix for the best performing feature vector classification approach based on AR-LS.

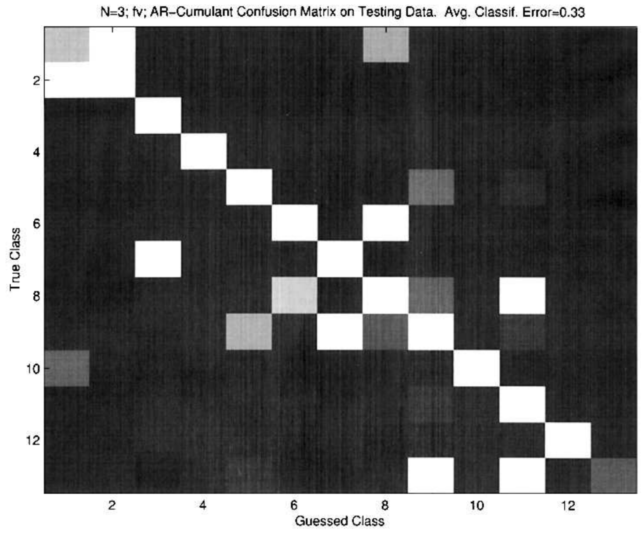

Fig. 8. Confusion matrix for the best performing feature vector classification approach based on AR-cumulant.

V. Performance Analysis

For the AR coefficient approach, we applied the neural network training strategy to both LS and cumulant approaches for each of the possible AR model orders. As expected, we found the two approaches to have similar performance. The best-performing AR-LS classifier used an AR model of order 2, getting a training cross-validation classification accuracy of 86.4%. Testing classification accuracy dropped to 65.1%. The best-performing AR-cumulant classifiers had 87.9% training accuracy, and 67.1% testing accuracy (model order 3). Figs. 7 and 8 shows the corresponding confusion matrices [24] on the testing data. If i is the row, and j is the column, then the brightness of location (i, j) indicates how often the RFB NN guessed that class j (the true class) was class i (the guessed class). If a classifier correctly identified everything, all the squares on the diagonal would be completely white, and the rest would be black.

The neural network performed the best with the low model orders; this is likely because it was able to generalize better over

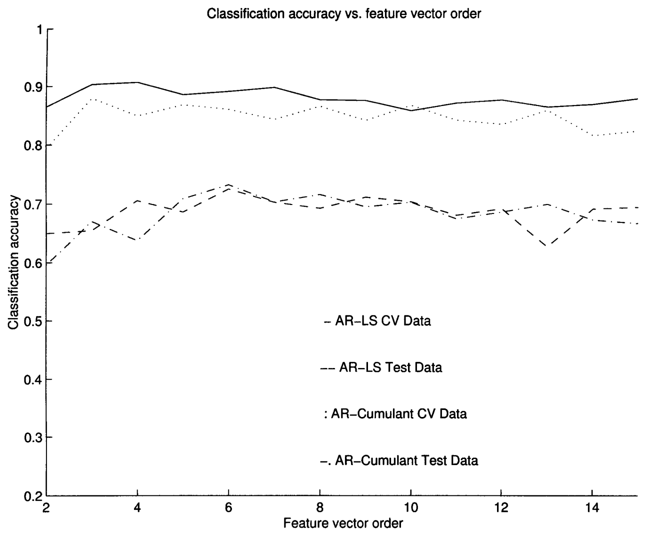

Fig. 9. Classification accuracy versus feature vector model order for both AR-LS and AR-cumulant approaches.

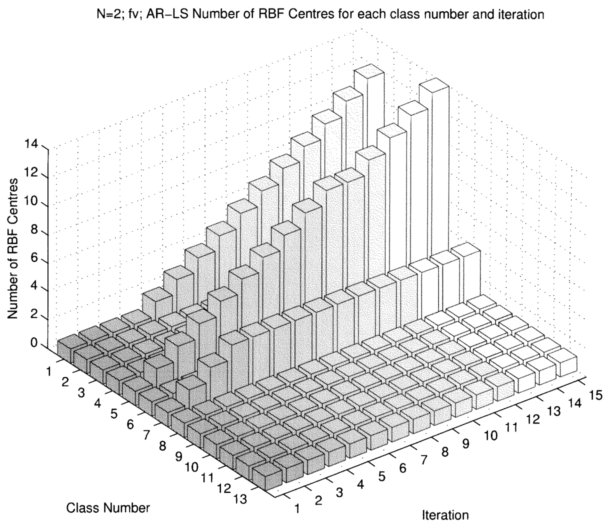

Fig. 10. Number of RBF centers for each class versus iteration for the AR-LS feature vector classification approach.

these smaller subspaces. Unfortunately, it means that "overlap" of classes occurred in higher dimensions as well. The higher model orders did have comparable performance to the lower model orders. Fig. 9 shows the neural network classification performance over all the model orders tested. Apparently, a higher model order cannot increase the separability of these audio radar signals.

We tracked the adaptation of the best-performing AR-LS classifier toward lower misclassification rates. Fig. 10 shows the growth of the neural network for each iteration, and Fig. 11 shows the corresponding decrease in cross-validation training error. We can see that centers get allocated to the trouble classes, until they are no longer trouble classes. For example, the number of centers allocated to class 7 grows after the first two iterations; after that, class 1 is given higher priority. On the other side, we observe from Fig. 11 that after the third iteration, the misclassification error caused by class 7 has lowered significantly. It is worth pointing out that misclassification rates of nontrouble classes drop as the iterations pass. This is usually the case because those nontrouble classes are misclassified as trouble classes. So, when there is better representation for the trouble classes, all-round misclassification drops. We also note that it took 15 iterations before stagnation occurred; this is good because the training error dropped fast enough. Over the iterations, the training error dropped from 27.5% to 13.6%.

For the prediction approach, we had a total of four different combinations of results: predictors were linear or nonlinear, and classification decisions were based on $arg(min(\cdot))$ or a neural network. It turned out that all four combinations had very similar performance to each other. The confusion matrices for each of the combinations are shown in Figs. 12 to 15 respectively.

The linear predictors with $arg(min(\cdot))$ decisions gets a training classification accuracy of 68.3%, and testing classification accuracy of 61.1%. These numbers have significant implications. It means that even with error being averaged over 29 predictions, predictors trained on some classes outperform predictors trained for the true class a large proportion of the time. This is due to the similarity between the classes. For example, an incoming class 1 signal is always classified as a

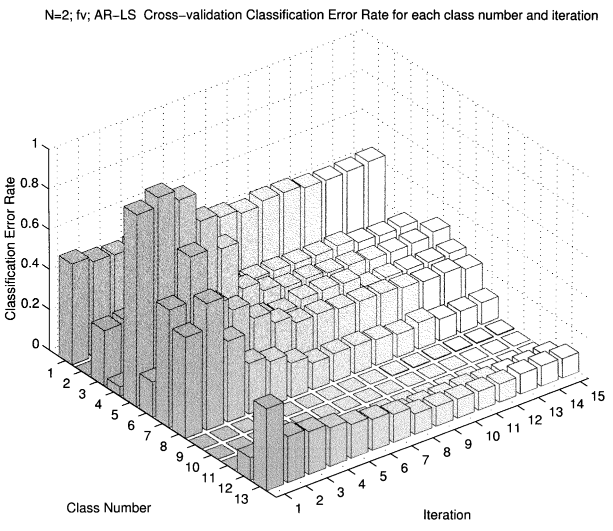

Fig. 11. Cross validation training error for each class versus iteration using the AR-LS feature vector classification approach.

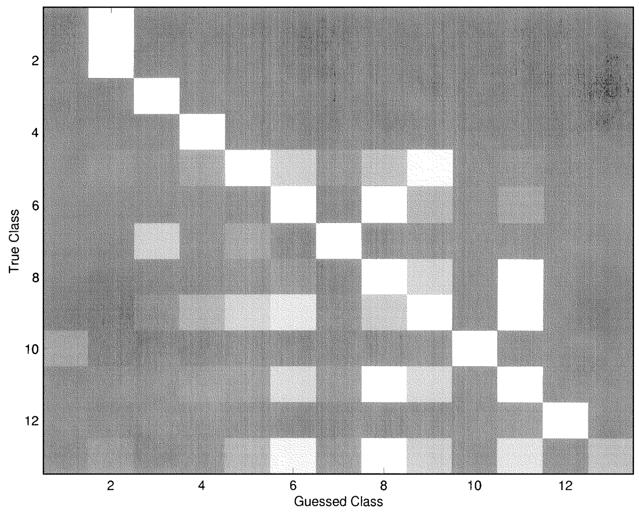

Fig. 12. Confusion matrix for linear predictor with classification decision based on predictor with lowest prediction error.

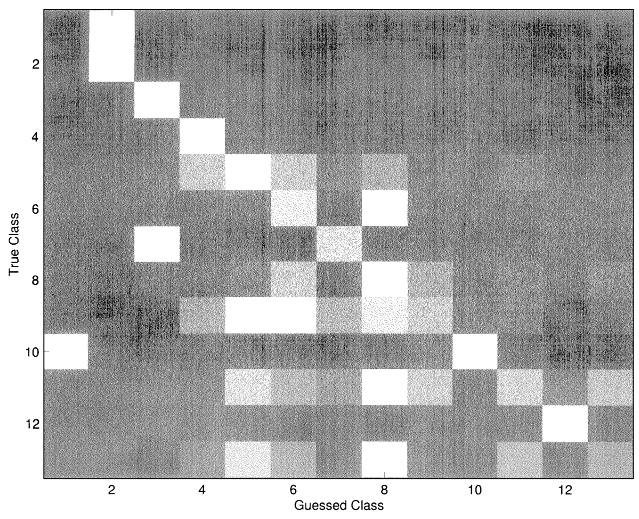

Fig. 13. Confusion matrix for linear predictor with classification decision based on a RBF network.

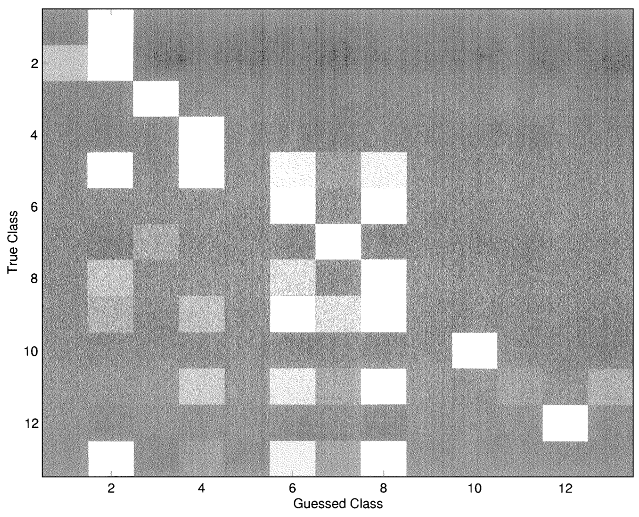

Fig. 14. Confusion matrix for neural network predictor with classification decision based on predictor with lowest prediction error.

class 2 signal (see Fig. 12, row 1, column 2), because the class 2 signal is quite similar to the class 1 signal (see the prediction errors of predictors 1 and 2 in Fig. 6).

With neural network decisions, the linear predictors achieved a training accuracy of 62.8%, and testing accuracy of 51.5%. It is surprising that the neural network decision module performed more poorly than the $arg(min(\cdot))$ module. Upon comparison of Figs. 12 and 13, we see that they are quite similar. However, there are some differences. For example, when the true class is 7, it is misclassified as a variety of different classes. When the true class is 10, it is misclassified as class 1. These problems do not happen with the $arg(min(\cdot))$ module.

The nonlinear predictors with $arg(min(\cdot))$ decisions get a training classification accuracy of 62.3%, and testing classification accuracy of 55.5%. This combination actually performs worse than the linear predictors with the same classification decision module. Upon re-examining the feature vectors generated by the nonlinear predictors, we realize that there are more occurrences of low prediction errors compared to the

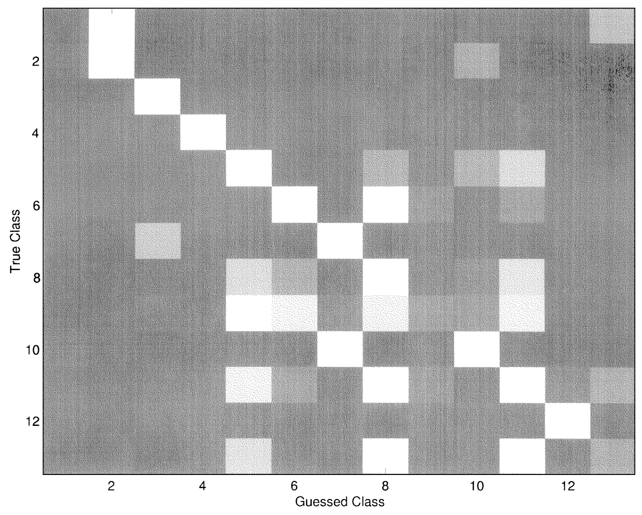

Fig. 15. Confusion matrix for neural network predictor with classification decision based on a neural network.

linear predictor feature vectors. Therefore, a simple classification decision scheme such as $arg(min(\cdot))$ will more readily make misclassifications.

With neural network decisions, the nonlinear predictors get a training accuracy of 69.5%, and testing accuracy of 57.2%. This was the highest training classification accuracy, and second highest testing accuracy, in the prediction approach. It appears that the feature vectors created by the neural network predictors may be better distinguished with a decision trained to recognize the differences in features, rather than to just simply choose the location of the minimum error.

VI. Conclusion

In this paper, we have applied RBF neural networks to perform automatic classification of audio radar signals. We have investigated different classification approaches based on linear autoregressive (AR) feature extraction and time series prediction. The classification results of these approaches on the real audio radar signals are summarized in Table I. We have also evaluated the raw data classification approach, but it only has a classification accuracy of 59% and a testing accuracy of 56.8%. We found that the cumulant-based AR feature vectors performed the best, with a cross-validated training classification accuracy of 87.9%, and a testing accuracy of 67.1%. The misclassifications were largely due to the data itself. Many of the classes were quite similar, and the classifier would easily confuse between these. Examination of the low-order linear feature vectors revealed that classes actually overlapped. This indicates that a more effective feature extraction should be developed for these signals so that the distances between the feature classes can be increased. It should also be noted that although a successful classification rate of 67% may not be accurate enough for military application, the performance of the neural network classifier on this data set is quite impressive compared to that of human beings. We performed an experiment on classification of these signals using 40 human beings. The same training and testing sets were used, and we found that the human classification performance was only about 27% in classification accuracy. We expect that with a larger training

TABLE I
SUMMARY OF CLASSIFICATION RESULTS BASED ON THE AR FEATURE AND PREDICTION APPROACHES

APPROACH	Training accuracy (%)	Testing accuracy (%)
AR Feature Vector		
AR-LS	86.0	65.1
AR-cumulant	87.9	67.1
Prediction		
Linear predictors, *arg*(*min*(·)) decisions	68.3	61.1
Linear predictors, neural network decisions	62.8	51.5
Nonlinear predictors, *arg*(*min*(·)) decisions	62.3	55.5
Nonlinear predictors, neural network decisions	69.5	57.2

data base, the accuracy of the neural network classifier can be improved, just like a more experienced human operator can carry out this target recognition function more successfully than a less experienced one.

REFERENCES

[1] V. G. Nebabin, *Methods and Techniques of Radar Recognition*. Norwood, MA: Artect House, 1995.

[2] M. Moruzzis and N. Colin, "Radar recognition by fuzzy logic," *IEEE AES Syst. Mag.*, pp. 13–19, July 1998.

[3] G. Whittington, T. Spracklen, J. Haugh, and F. Faulkner, "Automated radar behavior analysis using neural network architectures," in *Proc. SPIE*, vol. 1965, 1993, pp. 44–59.

[4] C. E. Lin and K. L. Chen, "Automated air defence system using knowledge-based system," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 27, pp. 118–124, Jan. 1991.

[5] M. I. Skolnik, Ed., *Radar Handbook*. New York: McGraw-Hill, 1992.

[6] W. Chang, B. Bosworth, and G. C. Carter, "Results of using an artificial neural network to distinguish single echoes from multiple sonar echoes," *J. Acoust. Soc. Amer.*, vol. 94, no. 3, pp. 1404–1408, Sept. 1993.

[7] Y. Shimshoui and N. Intrator, "Classification of seismic signals by integrating ensembles of neural networks," *IEEE Trans. Signal Processing*, vol. 46, pp. 1194–1201, May 1998.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[9] V. Petridis and A. Kehagras, "A recurrent network implementation of time series classification," *Neur. Comput.*, vol. 8, pp. 357–372, 1996.

[10] W. Hsu and R. R. Sylvian, "Construction of recurrent mixtures models for time series classification," in *Proc. Int. Joint Conf. Neural Networks*, July 1999, pp. 1574–1579.

[11] E. Haselsteiner, "Time series classification using adaptive dynamic targets," in *Proc. IEEE Workshop Neural Networks for Signal Processing*, Aug. 1999, pp. 243–252.

[12] D. Sand, Ed., *On-Line Learning in Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[13] R. J. Mammone, X. Zhang, and R. P. Ramachandran, "Robust speaker recognition: a feature-based approach," *IEEE Signal Processing Mag.*, pp. 58–71, Sept. 1996.

[14] M. Khalil and J. Duchene, "Detection and classification of multiple events in piecewise stationary signals: comparison between autoregressive and multiscale approaches," *Signal Processing*, vol. 75, pp. 239–251, 1999.

[15] J. Jouny, F. D. Garber, and R. L. Moses, "Radar target identification using the bispectrum: a comparative study," *IEEE Trans. Aerosp. Electron. Systems*, vol. 31, pp. 69–77, Jan. 1995.

[16] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 21, pp. 67–72, Jan. 1975.

[17] N. Hazarika, A. C. Tsoi, and A. A. Sergejew, "Nonlinear consideration in EEG signal classification," *IEEE Trans. Signal Processing*, vol. 45, pp. 829–836, Apr. 1997.

[18] H. Herzel, D. Berry, I. Titze, and I. Steinecke, "Nonlinear dynamics of the voice: signal analysis and biomechanical modeling," *Chaos*, vol. 5, pp. 30–34, Jan. 1995.

[19] A. Kehagias and V. Petridis, "Predictive modular neural networks for time series classification," *Neur. Networks*, vol. 10, pp. 31–49, 1997.

[20] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*. Reading, MA: Addison-Wesley, 1991.

[21] A. Swami, J. Mendel, and C. L. Nikias, *Higher Order Spectral Analysis Toolbox*. Natick, MA: The Mathworks Inc., 1995.

[22] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[23] M. Prakash and M. N. Murty, "Growing subspace pattern recognition models and their neural network models," *IEEE Trans. Neural Networks*, vol. 8, pp. 161–168, Jan. 1997.

[24] G. T. Toussant, "Bibliography on estimation of misclassification," *IEEE Trans. Inform. Theory*, vol. 20, pp. 472–479, Mar. 1974.

Trent McConaghy (S'95–M'99) received the B.E. degree in electrical engineering (with great distinction), and the B.Sc. degree in computer science (with great distinction) from the University of Saskatchewan, Saskatchewan, Canada, in 1999.

He is a Co-Founder and Chief Scientist of Analog Design Automation, Inc. (ADA), Ottawa, Ontario, Canada, which develops intelligent systems software to enhance the productivity of analog and mixed-signal circuit designers. Prior to co-founding ADA, he was a Researcher at the Canadian Department of National Defence, where he worked on automatic classification of radar signals and on prediction of chaotic time series. He has eight patents pending, and has been an invited speaker at several conferences and universities. His main interests include real-world problems, computer-aided design of analog circuits, nonlinear regression, evolutionary algorithms, and optimization.

Mr. McConaghy received the 2001 Outstanding Young Alumni Award from the University of Saskatchewan in recognition of significant accomplishments since graduation.

Henry Leung (M'90) received the Ph.D. degree in electrical and computer engineering from McMaster University, Canada.

He is currently a Professor with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, Canada. He was with the Defence Research Establishment, Ottawa, Ontario, Canada, where he was involved in the design of automated systems for air and maritime multisensor surveillance. His research interests include chaos, computational intelligence, data mining, nonlinear signal processing, sensor fusion, smart home, intelligent transportation system, multimedia, and wireless communications.

Éloi Bossé received the B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from Laval University.

In 1981, he joined the Communications Research Centre, Ottawa, Ontario, Canada, where he worked on radar signal processing, high resolution spectral analysis, and radar tracking in multipath. In 1988, he was transferred to the Defence Research Establishment, Ottawa (DREO) and, in 1992, he was transferred to the Defence Research Establishment, Valcartier (DREV), Quebec, Canada, where he worked on data and information fusion. He has published over 80 papers in journals and conference proceedings. He now heads the Decision Support Systems Section, DREV. His current interests include data and information fusion, reasoning under uncertainty, neural networks, and evidential and possibilistic theories.

Vinay Varadan received the M.S. degree from the Department of Electrical and Computer Engineering, University of Calgary, Calgary, Canada, in 2001. He is currently pursuing the Ph.D. degree in electrical engineering at Columbia University, New York, NY.

From 1998 to 1999, he served as a Software Engineer with Philips Software Centre, Bangalore, India. His research interests include computational biology, nonlinear signal processing, and evolutionary algorithms.