Principal Feature Classification

Qi Li and Donald W. Tufts

***Abstract*—The concept, structures, and algorithms of principal feature classification (PFC) are presented in this paper. PFC is intended to solve complex classification problems with large data sets. A PFC network is designed by sequentially finding principal features and removing training data which has already been correctly classified. PFC combines advantages of statistical pattern recognition, decision trees, and artificial neural networks (ANN's) and provides fast learning with good performance and a simple network structure. For the real-world applications of this paper, PFC provides better performance than conventional statistical pattern recognition, avoids the long training times of backpropagation and other gradient-decent algorithms for ANN's, and provides a low-complexity structure for realization.**

***Index Terms*—Statistical pattern recognition, neural networks, decision trees, discriminant analysis, classification, signal processing.**

I. Introduction

A PRINCIPAL FEATURE is a discriminant function which is intended to provide the maximum contribution to correct classification using a current training data set.

Principal feature classification (PFC) is based on a sequential procedure for finding principal features. After a new principal feature is found, the correctly classified vectors of the current training data are removed so that the next principal feature can best contribute to improve classification, rather than redundantly reclassify some of the already well-classified training vectors. PFC can also be considered as a nonparametric statistical approach for classification.

The procedure for finding principle features in PFC is analogous to a method for sequentially finding principal-component basis vectors [1]. Successive determination of principal features and associated, successive pruning of the training data are naturally different from the analogous steps for the principal components used in statistical pattern recognition because the PFC criterion, namely improvement in classification performance, is different from the mean-squared fitting error criterion of principal components. PFC provides a systematic method to sequentially select subsets of training data to compute the features, which was unsolved in a statistical feature approach [2].

A lot of work has been done in feature extraction and selection using both neural networks and traditional approaches (see [3] for a comparative study). Feature extraction focuses on reducing the dimensions of data; PFC focuses on removing subset of classifiable training data although feature extraction can be included in a PFC design without much additional calculation. PFC is suitable for the training problems with large data sets.

PFC does not need gradient-descent algorithms which have the local minimum and long training time problems as in backpropagation and many training algorithms; PFC does not need a backward node "pruning" (we use the term simplification in this paper for node pruning) as in a classification and regression tree (CART) [4] and neural tree network [5], or a node "pruning" by retraining (see [6] for a survey). PFC can be implemented either as a decision tree or as a neural network. For the above reasons, PFC can be considered as a fast and efficient algorithm for both neural network and decision tree design for classification.

Manuscript received January 12, 1996; revised June 26, 1996.

Q. Li was with the Department of Electrical and Computer Engineering, University of Rhode Island, Kingston, RI 02881 USA. He is now with Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974 USA.

D. W. Tufts is with the Department of Electrical and Computer Engineering, University of Rhode Island, Kingston, RI 02881 USA.

Publisher Item Identifier S 1045-9227(97)00229-4.

Example 1—An Illustrative Example for the Design Procedure

We use the two labeled classes of artificial training data in Fig. 1(a) to better specify the procedure of finding principal features and pruning the training data. We sequentially find principal features and associated hidden nodes at each stage by selecting the best of the following two methods for choosing the next feature: 1) Fisher's linear discriminant analysis (LDA) [1], [7], [8] and 2) *maximal signal-to-noise-ratio (SNR) discriminant analysis* (see Section III-A). A special method for determining multiple thresholds associated with these features has been developed by us [9] and is used in evaluating the effectiveness of candidate features. Although more complicated features, such as those from multivariate Gaussian [1], multivariate Gaussian mixture [10], or radial basis functions [11] can be used, the above two features are simple, complementary, and efficient.

In the first step, we use all the training data in the input space of Fig. 1(a) to find a principal feature. In this step, Fisher's LDA gave a good result. The corresponding feature can be calculated by an inner product of the data vector with the LDA weight vector. The hyperplanes perpendicular to the vector are shown in Fig. 1(b). It is important to note that multiple threshold values can be used with each feature. Then, the data vectors which have been classified at this step of the design procedure are pruned off. Here, two threshold values have been used. Thus the unclassified data between the two corresponding hyperplanes are used to train the second hidden node. The residual training data set for the next design stage is shown in Fig. 1(c). This is used to determine the second feature and second hidden node. Since the mean vectors of the two classes are very close now, Fisher's LDA does not give a satisfactory principal feature. In the second hidden node

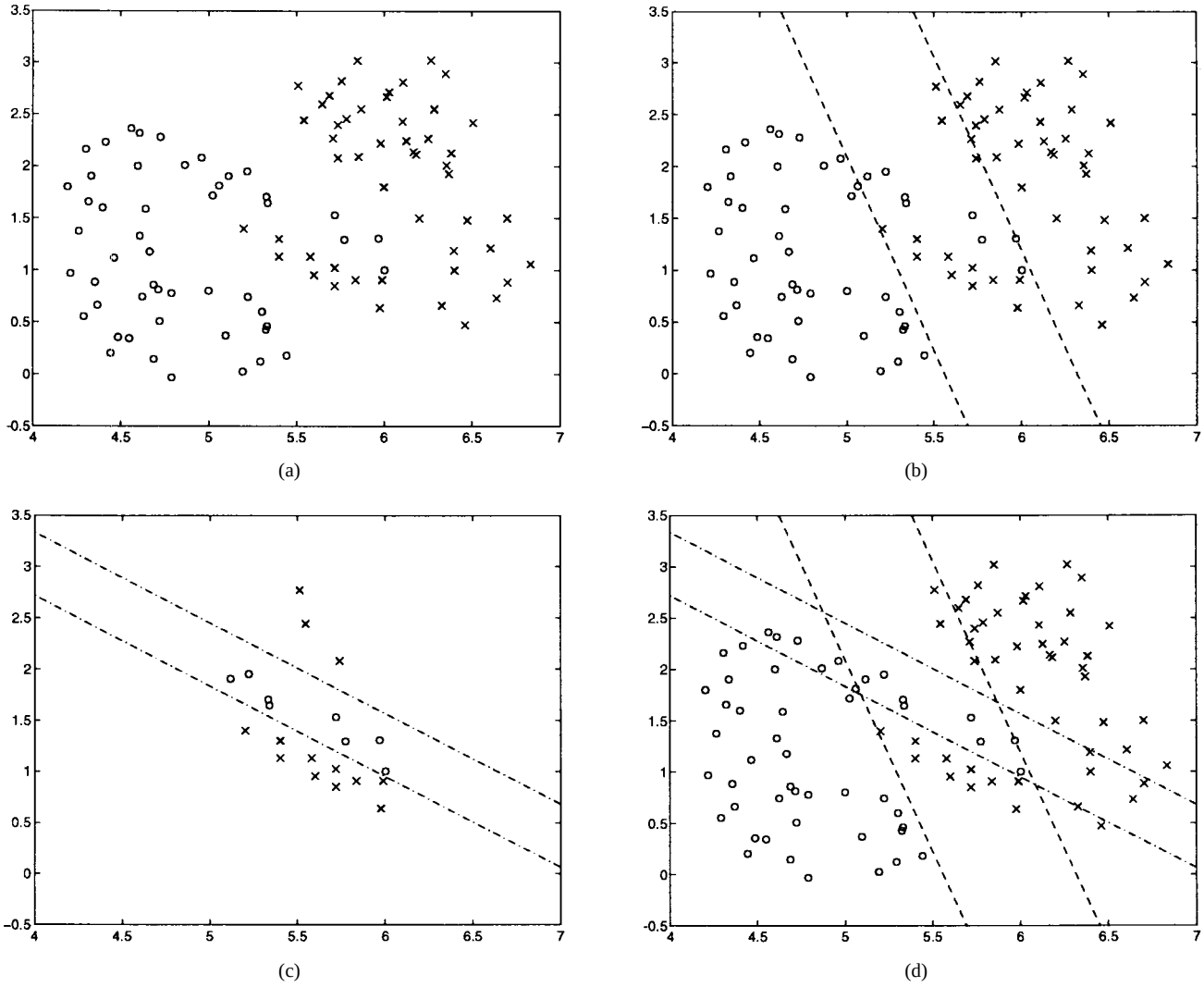


Fig. 1. (a) The original training data of two labeled classes which are not linearly separable. (b) The hyperplanes of the first hidden node (LDA node). (c) The residual data set and the hyperplanes of the second hidden node (SNR node). (d) The input space partitioned by two hidden nodes and four thresholds designed by the PFC method.

design, maximal SNR analysis provides a better candidate for a principal feature and the threshold-setting procedure in [9] gives us two associated hyperplanes which are also shown in Fig. 1(c). The overall partitioned regions are shown in Fig. 1(d). All of the training data vectors have now been correctly classified. The size of training data, the performance specifications, and the need to generalize to new, test data influence the threshold settings and the stopping point of the design.

For this simple classification problem, the backpropagation (BP) training method [12] takes hundreds of seconds to hours, and one still does not get satisfactory classification using a multilayer perceptron (MLP) network with five hidden nodes and one output node, sum-squared error (SSE) $= 4.35$. The radial basis function network (RBF) with common kernel functions [12] can converge to an acceptable performance in 35 s, but it needs 56 nodes, SSE $= 0.13$. On the same problem, the principal feature classification only takes 0.2 s on a same machine and needs only two hidden nodes in a sequential implementation, Fig. 5(b). The performance of principal feature network (PFN) is better than both BP and RBF, SSE $= 0.00$. □

II. The Implementation of a Principal Feature Network

A principal feature network is an artificial neural network (ANN) in which each hidden node computes the value of a principal feature and compares this value with multiple threshold values. A parallel implementation of the PFN is shown in Fig. 2. The outputs of the hidden-layer are binary words. Each word represents one partitioned region in the input data space. Each class may have more than one hidden-layer word. The outputs of the output-layer are binary words too, which are logic functions of the hidden-layer words, but each class is only represented by one unique output binary word.

Each hidden node threshold is labeled to one class, and the associated hyperplane partitions the input space into *classified* and *unclassified regions* for that class. All or nearly all of the training vectors within a classified region belong to the

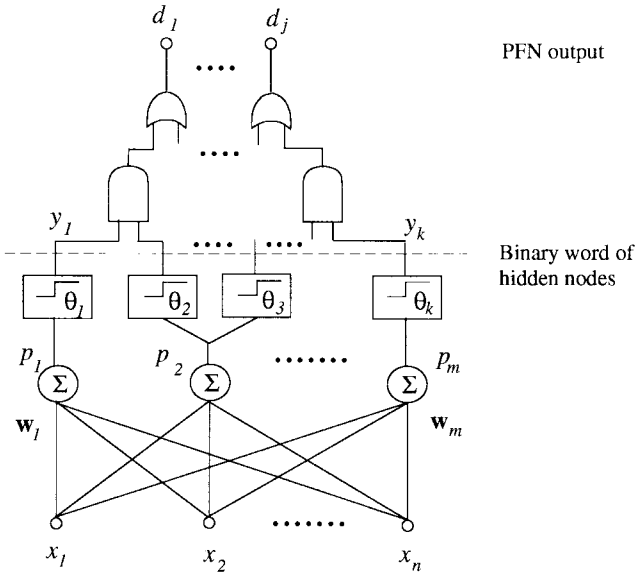


Fig. 2. A parallel implementation of PFC by a PFN.

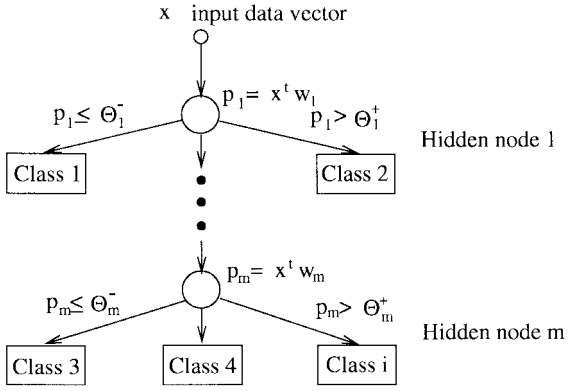
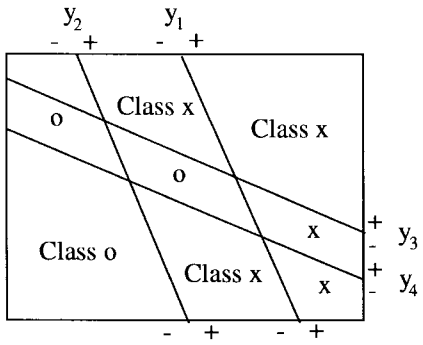
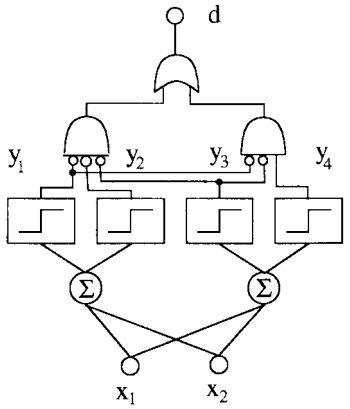


Fig. 3. A sequential implementation of PFC by a principal feature tree.



(a)



(b)

Fig. 4. (a) Partitioned input space for parallel implementation. (b) Parallel implementation.

labeled class. The corresponding unclassified region includes the training vectors which have not been correctly classified yet. Generally speaking, each *hidden node* is a subclassifier in the PFN. It classifies part of the training vectors and leaves the unclassified vectors to other hidden nodes.

This interpretation leads us to the decision-tree implementation of Fig 3. We note that each hidden node can have multiple thresholds, thus more than one classification decision can be made in one node of the tree, e.g., Fig 5(b). In sequence, the hidden nodes will be evaluated in the order that they are trained. Since each hidden node binary threshold output has been associated with one class in training, the sequential calculation stops as soon as a decision can be made. Since the first few hidden nodes are designed using the highest density regions in the input training space, it is very likely that a decision can be made early in the procedure. This is an advantage over multilayer perceptron (MLP) networks and conventional decision trees.

Example 1 (Continued)—Parallel and Sequential Implementations

Parallel and sequential implementations of the designed PFC are shown in Fig. 4(b) and 5(b). Corresponding partitioned input spaces are shown in Fig. 4(a) and 5(a). For details on parallel and processor-array implementations, please refer to [9] and [13].

III. Hidden-Node Design

A. Maximum SNR Hidden Node Design

When the mean vectors of training classes are far enough apart, Fisher's node is effective. However, when the mean vectors of the training classes are too close, Fisher's LDA does not provide good classification. Then one can use the quadratic Gaussian discriminant [1], or the following simple discriminant which can be used for non-Gaussian data and often has almost the same discriminant capability as the quadratic Gaussian discriminant when the data vector are multivariate Gaussian. A proof was given by us in [14].

To design a robust quadratic node for possibly non-Gaussian data, we choose a weight vector $\boldsymbol{w}$ to maximize a discriminant SNR

$$J = \frac{\boldsymbol{w}^t \Sigma_l \boldsymbol{w}}{\boldsymbol{w}^t \Sigma_c \boldsymbol{w}} \tag{1}$$

where Σ_l is the covariance matrix calculated from Class l which has the largest eigenvalue among the eigenvalues calculated from the sample covariance matrices of each class, respectively, and Σ_c is the covariance matrix calculated from

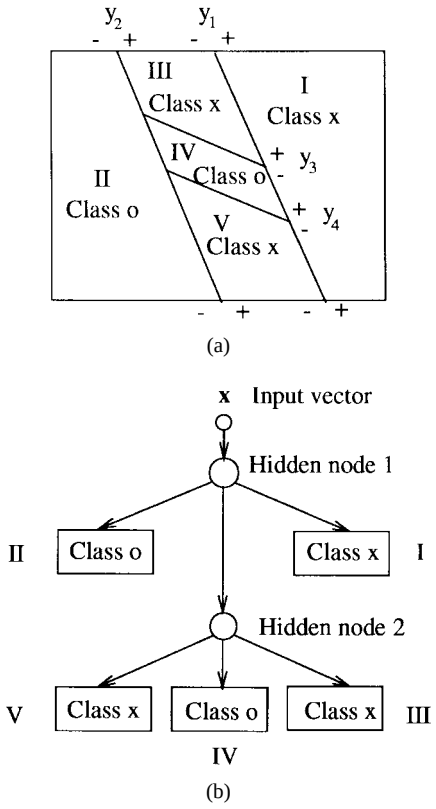


Fig. 5. (a) Partitioned input space for sequential implementation. (b) Sequential (tree) implementation.

pooled data of all other classes. The weight vector $\boldsymbol{w}$ can be determined by solving a generalized eigenvalue and eigenvector problem, i.e., $\Sigma_t \boldsymbol{w} = \lambda \Sigma_c \boldsymbol{w}$. The maximum SNR node has been used to classify overlapped Gaussian data [14].

B. Determining the Thresholds from Design Specifications

After a weight vector is obtained by LDA or by maximum SNR analysis, all current training vectors are projected to the weight vector. The histograms of projected data of all classes are evaluated. The classes on the far right and far left on the vector can be separated by determining thresholds. The thresholds and separated regions are then labeled to the separated classes.

A technique for determining thresholds from design specifications, i.e., performance requirements for every class, was developed by us and was applied in all of the examples and applications of this paper. Due to limited space, interested readers are referred to [9] for the details of this procedure.

IV. Simplification of the Hidden Nodes

Due to the PFN architecture and the training algorithm, pruning of the PFN hidden nodes is simpler than the pruning algorithms for MLP networks. We developed two kinds of pruning algorithms for different applications, lossless and lossy simplifications. *Lossless simplification* is for a minimal implementation; *lossy simplification* is for improving the ability of the network to generalize, that is, to perform well on new data sets. To avoid confusion with the data pruning described in the above sections, we use the term simplification. Generally speaking, lossy simplification is needed for most applications. Readers are referred to [9] for lossless simplification.

During PFN training, each threshold is labeled to a class which is associated with that threshold. We recall that each hidden node can have more than one threshold associated with separated classes. Also, the percentage of the training vectors of each class classified by each threshold in the sequential design are saved in an array. The array called the *contribution array* is used for simplification analysis. We use the following example to illustrate the details.

Application 1—Signal Recognition

In a real signal recognition application, a large set of multidimensional training vectors of 10 classes was completely classified by a PFN using 49 hidden nodes and 98 thresholds. The contribution of each threshold to its labeled class, in term of percentage of classification rate, is saved in a contribution array. The array was sorted and plotted in Fig. 6(a). From the Fig. 6(a), we can see that only a few of the thresholds have significant contribution to full recognition of their classes. The accumulated network performance in the order of the sorted thresholds is shown in Fig. 6(b). The more thresholds we keep, the higher the network accuracy we can obtain on the training data set, but keeping those thresholds which provide little contribution can affect the ability of the designed network to generalize to new data. In other words, using all of these thresholds may not lead to good performance on the test data set.

In the simplification procedure we seek to attain a desired network performance which comes from the design specifications. This value is used to prune thresholds. In this example, the desired network performance is 92% correct decisions. A horizontal dash-dot line in Fig. 6(b) marks the desired 92% accuracy. The line has an intersection with the curve of the accumulated network performance. By projecting the intersection onto the Fig. 6(a) as the vertical broken line in both Fig. 6(a) and (b), a necessary number of thresholds to meet the desired network performance can be determined. For this example, the first 38 thresholds in Fig. 6(a) can meet the 92% network accuracy as requested in the design specifications. Thus thresholds 39 to 98 in the sorted contribution array can be deleted.

If all of the thresholds associated with a hidden node have been deleted, then that hidden node should also be deleted. After this lossy simplification, the designed PFN has a performance of 91% on the training set and 88% on the test set using 31 hidden nodes and 38 thresholds. Thus, the performance on the test set is close to the performance on the training set.

V. Application 2—Land Cover Recognition From Multispectral Images

We applied the principal feature classification to recognize categories of land cover from three images of Block Island, RI, corresponding to three spectral bands—two visible and one

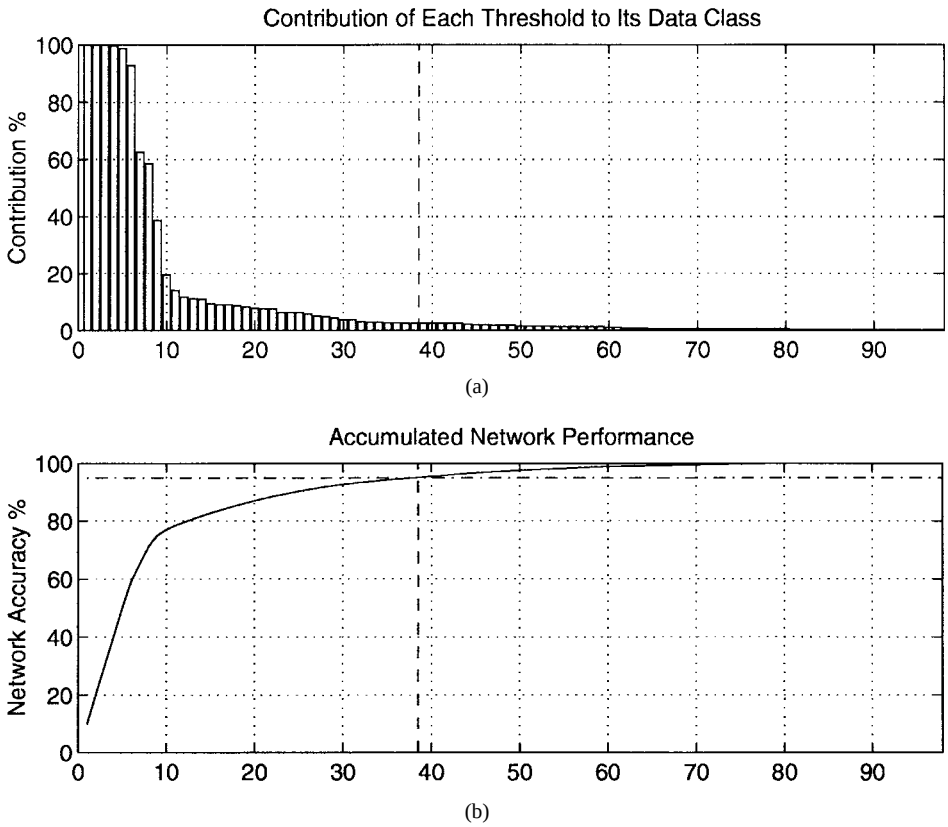


Fig. 6. (a) The sorted contribution of each threshold in the order of its contribution to the class separated by the threshold. (b) Accumulated network performance in the order of the sorted thresholds.

TABLE I
COMPARISON OF THREE ALGORITHMS IN THE LAND COVER RECOGNITION

Algorithms	Mflops	CPU Time (sec.)	No. of Nodes	Accuracy on Test Sets
PFN (proposed)	37.64	58	77	72%
MRBF [16]	221.93	518	490	60%
LDA [15]	–	–	–	55%

PFN: Principal feature network; MRBF: Modified radial basis function network; and LDA: Linear discriminant analysis.

infrared. Each complete image has 4591 × 7754 pixels. Each pixel has a resolution of 1.27 m and belongs to one of 14 categories of land covers.

The training data set is a matrix which is formed from a subset of pixels which have been labeled. Each row is one training vector which has nine feature elements associated with one pixel [15], and each of these vectors was labeled with one of 14 land cover categories. The nine features of data vectors consist of pixel intensity in the three color bands, three local standard deviations of intensity in a diameter of 10 m floating window around the row-designated pixel (one for each color), and three additional features from the side information of a soil database. These features are degree of local slope at the designated pixel, aspect of the slope at the designated pixel, and drainage class of soil. In [15], Duhaime identified the 14 categories of land covers for supervised training.

The computer experiments on the multispectral image features started by using backpropagation and RBF algorithms [12]. However, both of them did not get the needed classification results in a reasonable amount of time as estimated from their convergence speeds. Then the proposed PFN and a modified RBF algorithm [16] were applied to solve the problem. The experimental results are listed in Table I and compared with one another.

The MRBF method used a training data set of 140 sample vectors (limited by memory space), 10 from each category, and tested on a test data set of 700 samples, 50 samples from each category. It gets an average accuracy of 60% on the test

set for all of the 14 categories defined above. The training took 518 s central processing unit (CPU) time on a Sun Sparc IPX workstation. The PFN is trained by 700 training vectors since the PFN can be trained with much less memory space. It took only 58 s CPU time and reached an average performance of 72% on the same test set and on all 14 categories. (The performance is 65% if using the 140 sample set for training.) The performance of LDA was reported in [15]. It is 55% on an average of 11 categories out of all the 14 categories based on different training and test data sets. The simulation software was written in an interpretive language for both PFN and MRBF. The CPU time can be much less by using C or Fortran.

VI. Conclusions

Principal feature networks have been compared in experiments with popular neural networks, such as BP and RBF. It was also compared with many constructive algorithms [9], such as cascade-correlation architecture, decision tree algorithms, etc. Generally speaking, the PFC possesses the advantages of the constructive algorithms. It can get 100% accuracy on training set when it is needed. By applying multivariate statistical analysis in defining and training hidden nodes, the classifier can be trained much faster than gradient-descent or other iterative algorithms. The over-fitting problem which results from requiring too much classification accuracy on the training data is solved by appropriately pruning thresholds using the design specifications, and thus generalization to new test data can be realized by lossy simplification. PFC has been applied to real-world classification problems with large data sets. Compared with other algorithms, it has the same or better performance, needs much less CPU time in training, and uses simpler structures for implementation.

References

[1] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[2] S. S. Viglione, *Applications of Pattern Recognition Technology in Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, J. M. Mendel and K. S. Fu, Eds. New York: Academic, 1970.

[3] J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Trans. Neural Networks*, vol. 6, pp. 296–317, Mar. 1995.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[5] A. Sankar and R. J. Mammone, "Growing and pruning neural tree networks," *IEEE Trans. Comput.*, vol. 42, pp. 291–299, Mar. 1993.

[6] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, Sept. 1993.

[7] R. A. Fisher, "The statistical utilization of multiple measurements," *Ann. Eugenics*, vol. 8, pp. 376–386, 1938.

[8] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[9] Q. Li, "Classification using principal features with application to speaker verification," Ph.D. dissertation, Univ. Rhode Island, Kingston, Oct. 1995.

[10] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood training of probabilistic neural networks," *IEEE Trans. Neural Networks*, vol. 5, Sept. 1994.

[11] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, Mar. 1991.

[12] H. Demuth and M. Beale, *Neural-Network Toolbox User's Guide*. Natick, MA: MathWorks, 1994.

[13] Q. Li and D. W. Tufts, "Synthesizing neural networks by sequential addition of hidden nodes," in *Proc. IEEE Int. Conf. Neural Networks*, Orlando, FL, June 1994, pp. 708–713.

[14] Q. Li and D. W. Tufts, "Improving discriminant neural network (DNN) design by the use of principal component analysis," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, May 1995, pp. 3375–3379.

[15] R. J. Duhaime, "The use of color infrared digital orthophotography to map vegetation on Block Island, Rhode Island," M.S. thesis, Univ. Rhode Island, Kingston, RI, May 1994.

[16] Q. Li, D. W. Tufts, R.J. Duhaime, and P.V. Aug., "Fast training algorithms for large data sets with application to classification of multispectral images," in *Proc IEEE 28th Asilomar Conf.*, Pacific Grove, CA, Oct. 1994.