IMPROVING DISCRIMINANT NEURAL NETWORK (DNN) DESIGN BY THE USE OF PRINCIPAL COMPONENT ANALYSIS

Qi Li and Donald W. Tufts

Dept. of Electrical Engineering
University of Rhode Island, Kingston, RI 02881
tufts@ele.uri.edu and qli@ele.uri.edu

ABSTRACT

Investigations of the design of a Discriminant Neural Network (DNN) [1, 2, 3] have shown the advantages of sequential design of hidden nodes and pruning of the training data for improved classification and fast training time. The performance can be further improved by adding the capability of a nonlinear, principal component discriminant node. This type of hidden node is useful for separating classes which have common mean vectors and are overlapped on one other.

1. INTRODUCTION

A neural network architecture called a *Discriminant Neural Network* (DNN) and an associated design method were presented in [1, 2]. For several simulated and real classification problems, the DNN performances show advantages relative to several other training methods, such as linear discriminant analysis and modified radial basis functions, and the training is much faster than backpropagation and radial basis function networks [1, 2, 3].

The concept of DNN design is based on sequential discriminant analysis and pruning of training data. We sequentially design hidden nodes one by one. In each hidden node design, a corresponding discriminant node or a small set of discriminant nodes and associated threshold values are computed to provide best incremental improvement in classification. Then a subset of the training data which has been well-enough classified by the current hidden node is pruned and only the residual subset of training data is carried over to design the succeeding hidden nodes.

In this paper we improve the design procedure by providing an additional discriminant option for the design of each hidden node, or a small set of hidden nodes.

As shown in Figure 1, the DNN has two layers, one hidden layer and one output layer. The hidden layer consists of hidden nodes. Each hidden node forms one or several parallel hyperplanes to partition the input pattern space into regions. Several hyperplanes are used when we approximate a nonlinear discriminant. Each classification region is represented by a set of binary words which are the outputs of the hidden nodes.

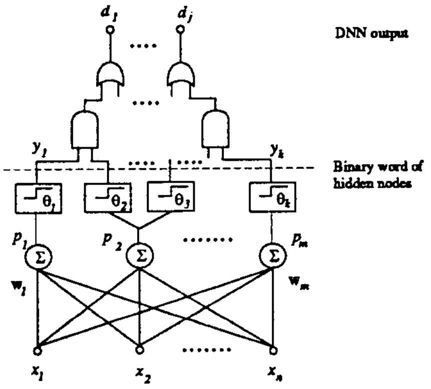

Figure 1: **One of implementations for a Discriminant Neural Network (DNN).**

The output of one of the biased hardlimiters of one hidden node in response to a given data vector $\mathbf{x}$ is $y = f(\mathbf{x}^t\mathbf{w})$ in which $\mathbf{x}^t\mathbf{w}$ is the inner product of the input vector $\mathbf{x}$ and weight vector $\mathbf{w}$. The biased hardlimiter function, $f(.)$, is specified by the formula,

$$f(\mathbf{x}^t\mathbf{w}) = \begin{cases} 1, & \mathbf{x}^t\mathbf{w} > \theta; \\ 0, & \mathbf{x}^t\mathbf{w} \le \theta, \end{cases} \tag{1}$$

in which θ is a threshold value which can also be considered as a bias for the argument of the nonlinearity. Several hardlimiters are allowed for any one of the hidden nodes.

2. GAUSSIAN DISCRIMINANT NODE

The single hidden node design algorithm is motivated by optimal multivariate Gaussian classification. When two training data populations Class 1 and Class 2 are

0-7803-2431-5/95 $4.00 © 1995 IEEE

described as multivariate Gaussian distributions with sample mean vectors and covariance matrices μ_1, Σ_1 and μ_2, Σ_2 respectively, the minimum-cost classification rule is given by [4, 5]:

$$Class1 : L(\mathrm{x}) \geq \theta; \quad Class2 : L(\mathrm{x}) < \theta; \qquad (2)$$

where x is an observed data vector or feature vector of N components and θ is a threshold determined by the cost ratio, the prior probability ratio, and the determinants of the covariance matrices, and

$$\begin{aligned} L(\mathrm{x}) &= \mathrm{x}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathrm{x} - 2(\mu_1^t\Sigma_1^{-1} - \mu_2^t\Sigma_2^{-1})\mathrm{x} \quad (3) \\ &= \sum_{i=1}^{N} \lambda_i |\mathrm{x}^t W_i|^2 - 2W_0\mathrm{x}, \end{aligned}$$

where $W_0 = (\mu_1^t\Sigma_1^{-1} - \mu_2^t\Sigma_2^{-1})$ and for $i > 0$, λ_i and W_i are the i'th eigenvalue and eigenvector for matrix $\Sigma_1^{-1} - \Sigma_2^{-1}$. We define the equation(3) as a *Gaussian Discriminant Node.* It's implementation is shown in Figure 2(a).

When the covariance matrices in (3) are the same, the first, quadratic term is zero, and the above classifier computes Fisher's linear discriminant. The general node becomes a Fisher's node, Figure 2(b). When the second term can be ignored, the above formulas only have the first quadratic term. Due to the sequential design procedure, in each DNN design step we only use the eigenvector associated with the largest eigenvalue or a small number of principal eigenvectors. Thus, we use the quadratic node as shown in Figure 2(c). The thresholded squaring function can be further approximated by two thresholds as in Figure 2(d).

3. PRINCIPAL COMPONENT (PC) HIDDEN NODE DESIGN

When the mean vectors of training classes are not too close to one other, Fisher's node is effective and has been presented in [1, 2] for hidden node design. However, when the mean vectors of the training classes are too close, Fisher's linear discriminant analysis does not provide good classification. Then we should use the quadratic node, or approximate it by a PC node.

3.1. Principal Component Discriminant Analysis

To design a quadratic node directly for non-qaussian data, we use the following criterion for an alternate discriminant: We choose a weight vector W to maximize a discriminant signal-to-noise ratio J.

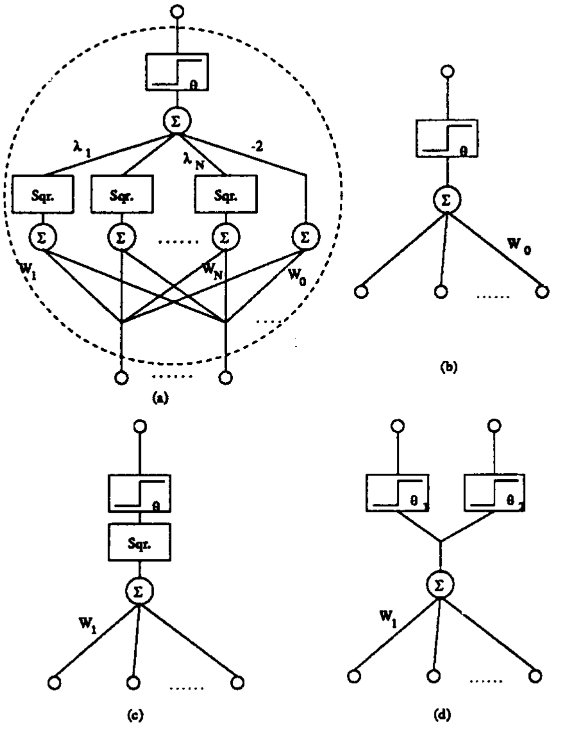

Figure 2: (a) A single gaussian discriminant node. (b) A Fisher's node. (c) A quadratic node. (d) An approximation of the quadratic node.

$$J = \frac{E\{(X_l W)^t (X_l W)\}}{E\{(X_c W)^t (X_c W)\}} = \frac{W^t \Sigma_l W}{W^t \Sigma_c W}, \qquad (4)$$

where X_l is a matrix of row vectors of training data from class l. X_c is the matrix of training data from all classes except X_l. The class l is the class which has the largest eigenvalue among the eigenvalues calculated from the data matrices of each class respectively. Σ_l and Σ_c are the estimated covariance matrices and W is the weight vector. In the case that mean vectors are same but not zero, the criteria still can be used. The weight vector W can be determined by solving the following generalized eigenvalue and eigenvector problem.

$$\Sigma_l W = \lambda \Sigma_c W, \qquad (5)$$

The eigenvector associated with the largest eigenvalue provide the maximum value of J. However, more than one weight vector can be selected to improve the discriminant analysis. In other words, more than one quadratic hidden node can be trained by solving the eigenvalue problem once.

We note that for non-gaussian classes it can be useful to replace the estimated covariance matrices in (4) by estimated correlation matrices.

3.2. Relation to the Optimal Gaussian Classifier

The PC of the previous section is intended for non-gaussian common-mean-vector classes, we prove as follows that it approximates the discriminant capability of a Gaussian classifier when the class data are from Gaussian distributions with zero mean vectors.

Let $W = \Sigma_c^{-1/2}V$ and $V = \Sigma_c^{t/2}W$. The equation (4) can be further written as

$$J = \frac{V^t\Sigma_c^{-t/2}\Sigma_l\Sigma_c^{-1/2}V}{V^tV} = \frac{V^t\mathbf{S}V}{V^tV}, \qquad (6)$$

where $\mathbf{S} = \Sigma_c^{-t/2}\Sigma_l\Sigma_c^{-1/2}$. The singular value decomposition (SVD) is $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^t$. The Λ is a diagnal matrix of eigenvalues. The maximum occurs when $V = \mathbf{U}_1$, where $\mathbf{U}_1$ is the eigenvector associated with the largest eigenvalue λ_1 of Λ. Thus, the weight vector for which J of (4) is a maximum is

$$W = \Sigma_c^{-1/2}\mathbf{U}_1. \qquad (7)$$

The classification functions of this quadratic node are

$$Class\ l : |\mathbf{x}^t\Sigma_c^{-1/2}\mathbf{U}_1^t|^2 > \theta' \qquad (8)$$

$$Class\ c : |\mathbf{x}^t\Sigma_c^{-1/2}\mathbf{U}_1^t|^2 \leq \theta', \qquad (9)$$

where θ' is a classification thresholds.

This can provide a good approximation to the performance of the gaussian classifier. The classification rule in (3) now can be written as

$$\begin{aligned} L(\mathbf{x}) &= \mathbf{x}^t(\Sigma_c^{-1} - \Sigma_l^{-1})\mathbf{x} \\ &= \mathbf{x}^t\Sigma_c^{-t/2}(I - \Sigma_c^{t/2}\Sigma_l^{-1}\Sigma_c^{-t/2})\Sigma_c^{-1/2}\mathbf{x}, \\ &= \mathbf{x}^t\Sigma_c^{-t/2}(I - \mathbf{S}^{-1})\Sigma_c^{-1/2}\mathbf{x}, \\ &= \mathbf{x}^t\Sigma_c^{-t/2}\mathbf{U}(I - \Lambda^{-1})\mathbf{U}^t\Sigma_c^{-1/2}\mathbf{x}, \\ &= \sum_{i=1}^{N}(1 - 1/\lambda_i')|\mathbf{x}^t\Sigma_\mathbf{c}{}^{-1/2}\mathbf{U}_i^t|^2, \qquad (10) \end{aligned}$$

Take the first component from formula (10). The classification function is same as the quadratic node in (8) and (9).

Furthermore, if we use two thresholds to approximate the square function in Figure 2(c), the classification rules of (8) and (9) becomes

$$Class\ l : \theta_1 \leq \mathbf{x}^t\Sigma_c^{-1/2}\mathbf{U}_1 \leq \theta_2 \qquad (11)$$

$$Class\ c : \mathbf{x}^t\Sigma_c^{-1/2}\mathbf{U}_1 > \theta_2,\ or\ \mathbf{x}^t\Sigma_c^{-1/2}\mathbf{U}_1 < \theta_1. \quad (12)$$

The implementation of (11) and (12) is shown in Figure 2(d). It is a normal DNN hidden node as defined in Figure 1. The thresholds, θ_1 and θ_2, can be determined from the histograms of the two classes of data after projection onto one dimension [2].

4. A DESIGN EXAMPLE

A design example is shown in Figure 3. The training data set in Figure 3(a) includes two classes of data both having Gaussian distributions with zero means, so they are overlapped on each other. The data of Class 1 has 1000 samples represented as "o", and Class 2 has 500 samples presented as "×". The test data set is shown in Figure 3(b). A gaussian classifier and a general node as in equations (3) and (11) was designed with the threshold $\theta = 0$. The classification boundaries are shown in Figure 3(b) also.

The data-adaptive design method proposed in Section 3 was applied to design a classifer with low misclassification rate for the same data set. We first designed a classifier using two quadratic nodes. The first quadratic node is designed as equations (8) and (9) with entire training data set. The determined threshold value is $\theta_1 = -4.52$. The classification boundaries of the designed first node is shown in Figure 3(c), then the well-enough classified portion of Class 1 was pruned from the training data set as shown in Figure 3(d). The second quadratic node is designed as the first one but using the residual data set. The classification boundaries of the second node are shown from Figure 3(d) with $\theta_2 = -8.34$. The partationed regions are shown in Figure 3(g).

When we removed the well-enough classified portion of the Class 2 from Figure 3(d), the residual data set in the center specifies the overlapped portion of the two classes. We define the overlapped region as an *uncertain region.* It defines a third class, Class 3, as the uncertain class. The uncertain class arise when the misclassification rate between Class 1 and Class 2 - the classes of interest would otherwise be larger than permitted for the application of interest.

The PC nodes are also applied in this example. The first PC node was designed using all the training data. The two hyperplanes associated with the two thresholds of the first node are shown in Figure 3(e). Then, the well-enough classified data was pruned, and the residual data set was used to design the second hidden node, Figure 3(f). The hyperplanes of the designed DNN classifier are shown with the testing data set in Figure 3(h). The partitioned region in the center is an uncertain region.

The confusion matrices of designed classifier are listed in Table 1. The traditional Gaussian classifier in Table 1(a) has high misclassification rates of 6.6% and 11.4%. The classifier designed by the data-adaptive method and using the quadratic nodes has the performance, in Table 1(b), on the misclassification rates of 0.2% and 0.4%. The performance of the DNN designed

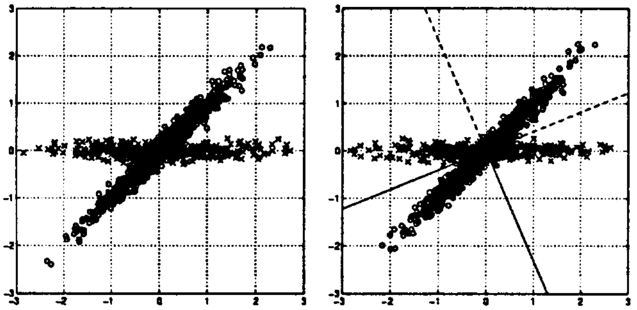

Figure 3(a) Two classes of training data. Figure 3(b) Test data set and the boundaries of a Gaussian classifier and a general node.

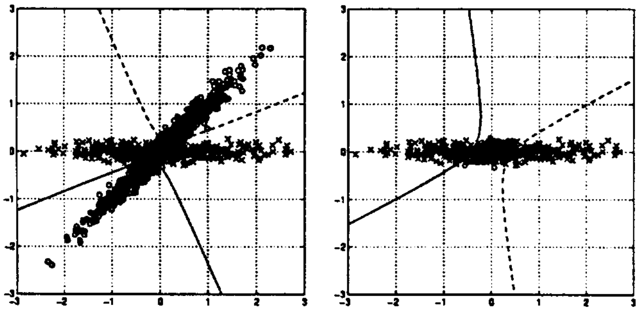

Figure 3(c) Design the first quadratic node with $\theta = -4.52$. Figure 3(d) Design the second quadratic node with $\theta = -8.34$.

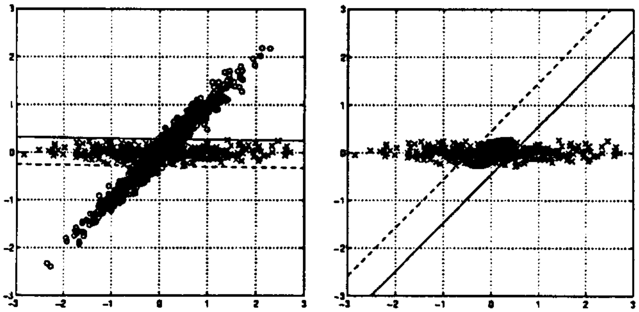

Figure 3(e) The classification boundaries of the first DNN node. Figure 3(f) The residual training data set and the classification boundaries of the second DNN node.

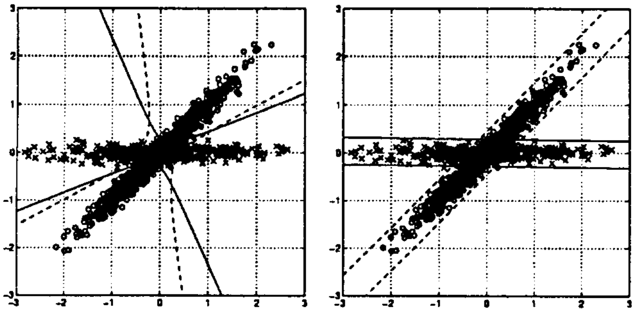

Figure 3(g) The test data set and the classification regions partitioned by quadratic nodes. Figure 3(h) The test data set and the classification regions partitioned by DNN classifier.

using two thresholds to approximate the square functions has the misclassification rates of 0.1% and 0.0%, Table 1(c), which are close to the performance of the quadratic nodes but the implementation is simpler. The low misclassification rates are obtained by reducing the classification rates, so the classification rates in (b) and (c) are lower than (a). However, it is useful in some applications which need very low misclassification rate.

Table 1. Confusion Matrices

	C1	C2
C1	93.4%	6.6%
C2	11.4%	88.6%

(a)

	C1	C2	Uncertain
C1	77.3%	0.2%	22.5%
C2	0.4%	61.4%	38.2%

(b)

	C1	C2	Uncertain
C1	72.3%	0.1%	27.6%
C2	0.0%	61.8%	38.2%

(c)

(a) The performance of the optimal Gaussian classifier and general node as in figure 3(b). (b) The performance of using two quadratic nodes as in Figure 3(g). (c) The performance of DNN using two linear nodes with two thresholds on each of them as in Figure 3(h).

5. REFERENCES

[1] Q. Li and D.W. Tufts, "Synthesizing neural networks by sequential addition of hidden nodes," Proc. IEEE International Conference on Neural Networks, pp. 708-713, Orlando, Florida, June 1994.

[2] Q. Li and D.W. Tufts, "Discriminant networks: a simple, effective, and rapidly trainable class of neural networks," Submitted to the *IEEE Trans. on Neural Networks*, February 1994.

[3] Q. Li, D.W. Tufts, R.J. Duhaime, and P.V. AugustFast, "Training Algorithms for Large Data Sets with Application to Classification of Multispectral Images," Proc. IEEE 28th Asilomar Conference, Pacific Grove, CA, October 1994.

[4] R.A. Johnson and D.W. Wichern, "Applied multivariate statistical analysis", pp. 470-530, New Jersey: Prentice Hall, 1988.

[5] L.L. Scharf, Statistical Signal Processing, Reading MA: Addison-Wesley, 1990.