专 题 汇 编

IJCNN' 有导师学习

Optimizing Multilayer Neural Networks using Fractal Dimensions of Time-series Data

Ikuo Matsuba, Hironari Masui, and Shingo Hebishima*

Systems Development Laboratory, Hitachi, Ltd.
1099 Ohzenji, Asao-ku, Kawasaki-shi 215, Japan
*Hitachi Information & Control Systems, Inc.
2-1 Omika-cho, Hitachi-shi 319-12, Japan

Abstract

A fractal dimension of time-series data is used to optimize the three-layer feed-back neural network which was proposed in the previous paper to detect an important time structure of time-series data and to predict a future sequence based on a current input sequence. Optimization means in a sense that the prediction error is minimized. A time interval giving the same fractal dimensions is used as an optimal size of the output layer. The number of input units is twice the number of output units. Furthermore, it is also found that reliability in prediction is determined empirically as a function of the fractal dimension.

1. Introduction

The representation of temporal knowledge and time correlation in multilayer neural networks presents a continuous challenge to researchers in the field of artificial neural networks. In the previous paper [1],[2], a feed-back neural network was proposed and it was found that this network is capable of detecting a time structure embodied in time-series data . In particular, a regularly organized time structure is found. This architecture is appropriate for a network in predicting a sequence based on a current sequence because it can easily respond according to the conditions of the order and frequency of the input sequence arriving at the network.

However, there remains a problem. The most important concern is the question of how to optimize the network to obtain the best performance. Provided a three-layer neural network, the number of hidden units can be determined by the information criterion such as AIC (Akaike's Information Criterion) to minimize the output (prediction) error using untrained data [1], [2]. A remaining question is what sizes of input and output layers are appropriate for detecting an inherent time structure and for predicting a future sequence with the smallest error. This problem is hard to be solved because a unique solution to this problem has not been obtained. The reason is that the performance of the network depends strongly on a statistical feature of the given time-series data. A fractal dimension is found to be the most suitable measure of the statistical feature. Furthermore, it is also found that reliability in prediction is determined empirically to be an approximately linear function of the fractal dimension. The specific example considered here consists of time-series data taken from economic indices.

0-7803-0559-0/92 $3.00 © 1992 IEEE

2. Fractal Dimensions of Time-series Data

The fractals considered here are sets of states generated by a system whose dynamical behavior is governed by nonlinear equations. Restricting the discussion to one dimension, a random process $x(t)$, i.e., a function x of time t whose values are random variables such as $x(1)$, $x(2)$. If $x(t)$ contains equal power for all frequencies f, this process is a white noise. If the spectral density $S(f)$ is proportional to $1/f^2$, the usual Brownian motion is obtained. In general, the density proportional to $1/f^{\beta}$ corresponds to the fractal Brownian motion whose mean square increments described by

$$E[|x(t_2)-x(t_1)|^2] \sim |t_2-t_1|^{2H} \tag{2.1}$$

with the relationship $\beta=2H+1$. Here, E denotes an expectation value. In general, the fractal dimension D ($1 \le D \le 2$) is defined using β or H as:

$$D=2-H=(5-\beta)/2 \tag{2.2}$$

Recently, a very convenient method for calculating fractal dimensions was developed by Higuchi [4]. Let $x(t)$ be a given sequence in the interval ($1 \le t \le N$). The total length $\Delta_k(t)/t$ of this curve is calculated from

$$\Delta_k(t)=(N/L_t)^{-1}\Sigma_{i=1}^{L_t}|x(k+it)-x(k+(i-1)t)| \tag{2.3}$$

with $L_t=[(N+1-k)/t]t$, where the notation $[z]$ is an integer not larger than z. See Fig. 1 for the definition of $\Delta_k(t)$. Then, we calculate an average value $\Delta(t)$ of $\Delta_k(t)$ with respect to k. If $\Delta(t)/t$ scales as t^{-D}, then D gives a fractal dimension of this time-series data. If the spectral density shows a power law behavior, then the relationship (2.2) is satisfied. From the definition of $\Delta(t)$, the deterministic process gives $D=1$. It is clear that D approaches 2 as the degree of stochasticity increases.

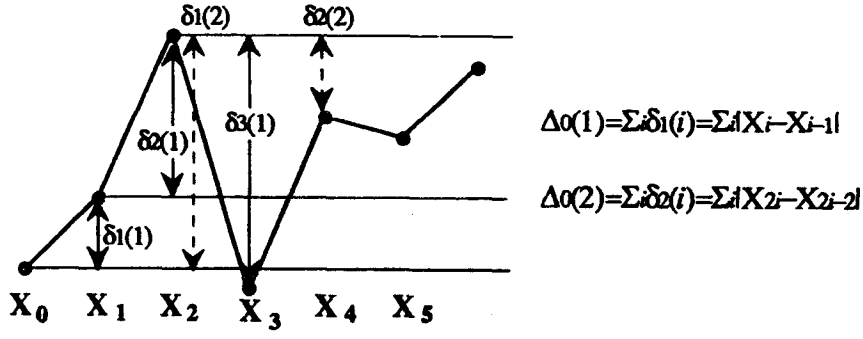


Fig. 1. Definition of $\Delta_k(t)$.

Let us first consider the following simple cases. A first example of the random process is generated by $x(t)=at+\xi$, where a is a positive constant and ξ is an additive Gaussian random variable with zero mean. Figure 2 shows the bahavior of this process and its corresponding fractal dimensions. An interesting fact is that there exist two distinct time regions: $1 \le t \le t_c$ and $t_c \le t$ where t_c is about 7. One corresponds to a random process (D=1.91), while the other represents a linear deterministic process (D=1.00). This implies that when the process is observed in a short time scale ($1 \le t \le t_c$), it is essentially random. On the contrary, this process is described by a deterministic process when we

observe it in a longer time scale ($t_c \leq t$). In other words, two distinct organized time structures are observed in each region. Therefore, it is unlikely to predict the sequence whose time interval is within t_c.

Another example using a moving average of an economic index is shown in Fig. 3. Here, also, are two distinct time regions. In this case, however, there is a different situation in which D=1.13 in a short time scale ($1 \leq t \leq 20$) and D=1.91 in a longer time scale ($30 \leq t$). Therefore, it will be possible to predict the sequence whose time interval is within t_c.

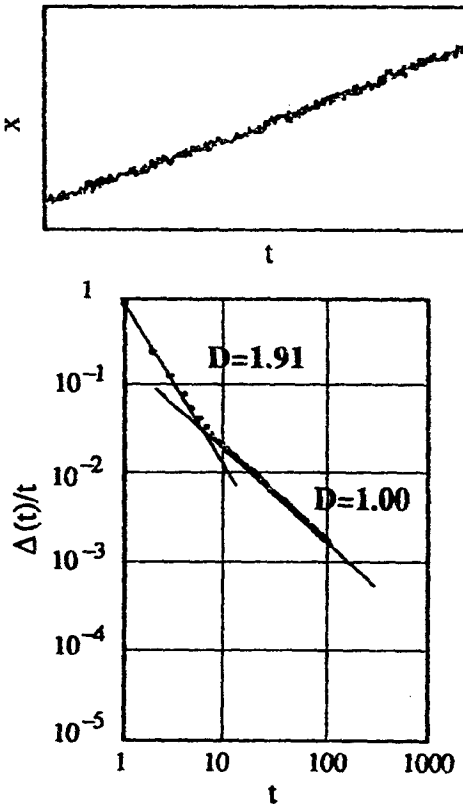


Fig. 2. Fractal dimensions of random time-series data.

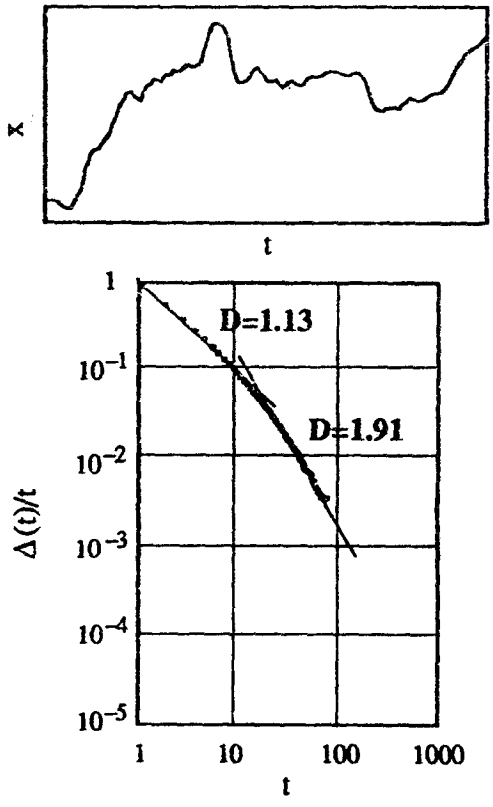


Fig. 3. Fractal dimensions obtained from a moving average of an economic index.

3. Optimal Sizes of Input and Output Layers

Neural networks are optimized with respect to the numbers of input and output units. To explain explicitly using a concrete example, a prediction problem is considered here. The purpose is to predict the sequence $x_{n+1} \sim x_N$ based on the current input sequence $x_1 \sim x_n$. Let x_0 denote an average of the data used in the learning period. To normalize the input and output data in the interval [0, 1], they are scaled as $[1+\tanh\{(x_t-x_0)/A\}]/2 \rightarrow x_t$, where A is a normalization constant. In practice, A is chosen so that the scaled data are in the interval [0.1, 0.9]. Because of the possibility that the future sequence is beyond the largest data sampled in the learning period, linear transformation is not acceptable.

It was found that the feed-back neural network having a structure as shown in Fig. 4 is capable of detecting an important time structure of the given time-series data [1], [2]. However, there remains

the problem of determining the network size, especially the number of input and output units.

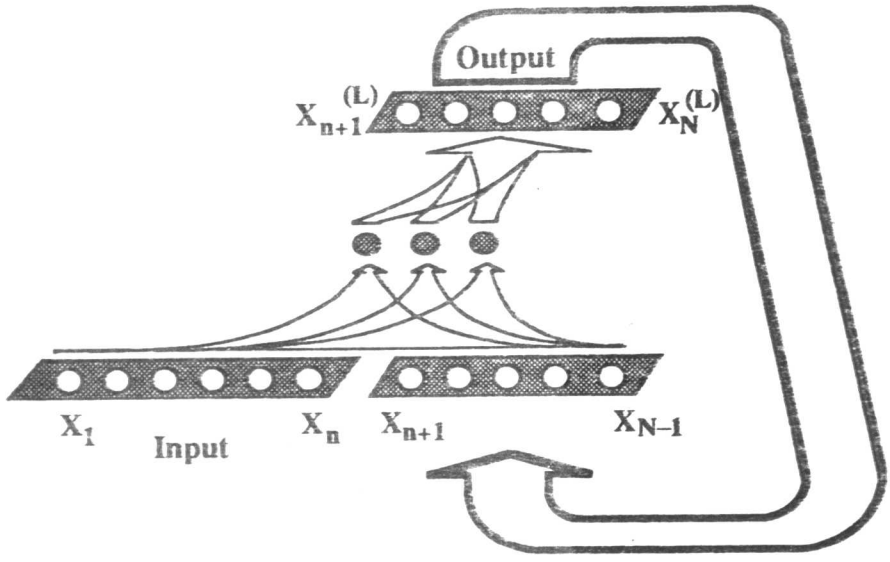
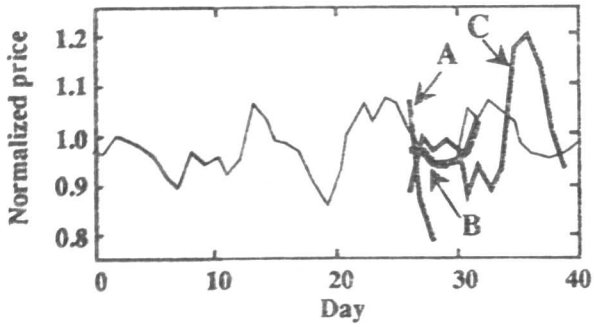


Fig. 4. Schematic diagram of a feed-back neural network for time-series data.

It is clear that the optimal size is highly dependent on the stochastic nature of the given time-series data. For example, when the time-series data shown in Fig. 3 is used, it is likely that we cannot predict a long-term sequence with an interval longer than t_c. Therefore, the number of output units must be smaller than this value to obtain reasonable prediction performance. Eventually, it is found that if the number of output units is taken to be t_c, the normalized prediction error defined by

$$1/t_c \Sigma_{t=1}^{tc} \{x_t - x_t^{(L)}\}^2 \qquad (3.1)$$

is minimized. Here, L is 3 for the three-layer network. Similar consideration is given to the the input layer whose size is found to be twice the number of output units, and the best prediction is thus obtained.



	Number of neurons			Error
	Input	Hidden	Output	
A	6	13	3	0.239
B	14	13	7	0.133
C	28	13	14	0.454

Fig. 5. Prediction errors for three different network sizes.

Figure 5 shows three numerical examples of prediction obtained by varying the number of input and output units. Here, the number of hidden units, 13, is determined by applying the AIC procedure [3]. In the left figure, x_t divided by the moving average is shown as a function of t (day). It is clear that the optimal size (Case B) determined from the fractal dimension gives the minimum prediction error as expected.

4. Application to Economic Index Prediction

While the optimal size of the hidden layer is determined by the AIC method, the optimal sizes of input and output layers are given by the fractal dimensions of the time-series data. In this section, some numerical examples are given for seeing the effectiveness of the proposed optimization procedure. Time-series data is sampled from an economic index (Nikkei average), and the deviation from its moving average is shown in Fig. 6. The fractal dimension is found to be D=1.57 within $1 \leq t \leq 10$. In simulation, therefore, the sizes of input, hidden, and output layers are taken to be 20, 8, and 10 in the learning phase, respectively. The number of back-propagation learning steps is 5,000 using 20 training sequences, and the learning coefficient is 0.5 without the momentum term.

Among 32 cases of prediction, there were good predictions 19 times using the present optimized feed-back neural network. That is, the prediction rate was 59%. Figure 6 shows 7 numerical examples of prediction. In this figure, the bold curves are predicted sequences, while the real sequential data are represented by the solid curve. Most predicted sequences follow the real data with only small errors. A satisfactory result is thus obtained.

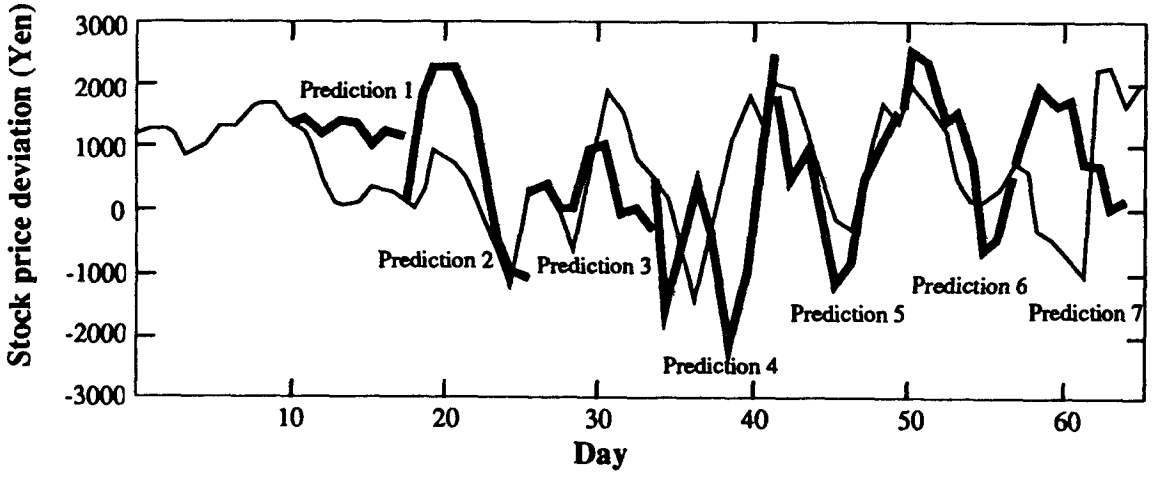


Fig. 6. Examples of prediction using an economical index sampled from the Nikkei average.

5. Evaluation of Reliability in Prediction

Now that the optimization procedure is established for time-series data. From the above discussions and the simulation results, it seems that the prediction rate depends strongly on the fractal dimension of the time series data. If a relationship between the prediction rate and the fractal dimension exists, reliability in prediction can be evaluated before predicting. Since no theoretical consideration is given right now, we have determined this relationship empirically. Each predicted sequence is categorized into 5 patterns according to its behavior as shown in Fig. 7.

Using several items of time-series data sampled from economic indices and natural phenomena, this relationship is found to be an approximately linear function of the fractal dimension as shown in Fig. 7. The lowest prediction rate is clearly 20% since an arbitrary single pattern randomly selected from five kinds of patterns is 1/5. This figure indicates that when the time-series data varies in a complicated manner, the corresponding prediction rate is low because of a high fractal dimension.

On the other hand, a low fractal dimension gives a high prediction rate since the time-series data varies in a rather smooth fashion.

When we predict the sequential data whose behavior is shown in Fig. 6, the prediction rate obtained is 59%. Considering that the fractal dimension of this time-series data is 1.57, the corresponding estimated prediction rate is 54.4% from the empirical relationship. This value should be compared to the 59% rate.

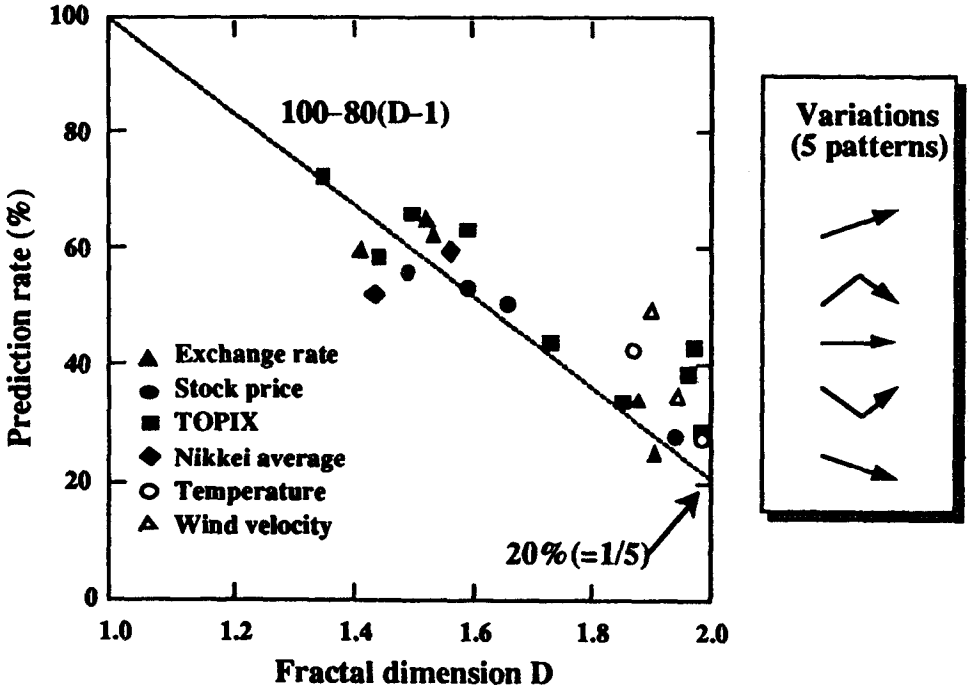


Fig. 7. Prediction rate as a function of the fractal dimension.

6. Summary

To summarize, fractal dimensions of time-series data were used to optimize neural networks. The fractal dimensions were also used to determine the number of input and output units. This method together with the AIC procedure to determine the number of hidden units is useful for optimizing a neural network for time-series data. Moreover, a relationship between a prediction rate and a fractal dimension was obtained empirically.

References

[1] I. Matsuba, "Neural sequential associator and its application to stock price prediction," IECON'91, Kobe, Japan, October, pp. 1476-1479, 1991.

[2] I. Matsuba, "Application of neural sequential associator to long-term stock price prediction," IJCNN'91, Singapore, November, pp. 1196-1202, 1991.

[3] I. Matsuba, "Feature detection by back-propagation," Suri-Kagaku (in Japanese), No. 338, pp. 31-37, 1991.

[4] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," Physica D, vol. 31, pp. 277-283, 1988.

Analysis of a Learning Algorithm for Neural Network Classifiers

Zhen-Ping Lo, Yaoqi Yu, and Behnam Bavarian
Department of Electrical and Computer Engineering
University of California, Irvine
Irvine, CA 92717

Abstract

In this paper we provide a convergence analysis of a learning rule which we derived for the adaptation of the neurons' synaptic weight vectors representing the prototype vectors of the class distribution in a classifier. The analysis also provides a theoretical foundation for the Kohonen learning vector quantization (LVQ1 and LVQ2) algorithms. The convergence of the learning rules are proved under certain conditions. More specifically, We show that the algorithm will converge to error free solution when the input patterns are linearly separable.

1 Introduction

In a previous paper[1], we derived three learning rules using the gradient decent method to adjust the decision surface to minimize the classification error. These learning rules resemble the LVQ2 algorithm. The reason why using the LVQ2 algorithm is better than the LVQ1 algorithm in the classifier design was also analyzed.

In this paper, convergence of the learning rules will be analyzed. The three learning rules generally will converge to the same results. Thus, we will only discuss one of them here. We will choose the case in which the weight vector is modified whenever a sample is misclassified.

The learning rule is given by

$$\mathbf{m}_1(k+1) = \mathbf{m}_1(k) - \alpha(\mathbf{x}_2{}^l - \mathbf{m}_1(k)) \tag{1}$$

or

$$\mathbf{m}_1(k+1) = \mathbf{m}_1(k) + \alpha(\mathbf{x}_1{}^l - \mathbf{m}_1(k)) \tag{2}$$

When input $\mathbf{x}$ belongs to class 2 but is misclassified as class 1, and $\mathbf{m}_2$ is fixed. Eq.(1) intuitively tells us to adjust the decision surface by adaptively moving $\mathbf{m}_1$ farther from the misclassified vector $\mathbf{x}$. When input $\mathbf{x}$ belongs to class 1 but is misclassified as class 2, and $\mathbf{m}_2$ is fixed, Eq.(2) intuitively tells us to adjust the decision surface by adaptively moving $\mathbf{m}_1$ closer to the misclassified vector $\mathbf{x}$.

The case 1 learning rule of Eq.(1) and (2) can be rewritten as

$$\mathbf{m}_1(k+1) = \mathbf{m}_1(k) + \alpha_{k+1} N_k(\mathbf{x}(k) - \mathbf{m}_1(k)) \tag{3}$$

where

$$N_k = N_k(\mathbf{x}(k), \mathbf{m}_1(k)) := \begin{cases} -1 & \text{if } \mathbf{x}(k) \in c_1(k) \cap c_2 \\ 1 & \text{if } \mathbf{x}(k) \in c_2(k) \cap c_1 \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{x}(k) \in c_i(k) \cap c_j$ means that $\mathbf{x}(k)$ is from class c_j but misclassified as c_i.

We will assume that the learning gain α_k satisfies the following conditions:

0-7803-0559-0/92 $3.00

$$\sum_{k=1}^{\infty} \alpha_k^2 < +\infty,$$

$$\sum_{k=1}^{\infty} \alpha_k = +\infty.$$

2 The Convergence of the Learning Rules

First let's consider the 1-dimensional case.

Theorem 1 *If the prototype vectors are initially labeled correctly* ($\mathbf{m}_i(0) \in c_i$), *then the learning rule of Eq.(3) converges to some limit.*

Proof: For convenience, we assume $\alpha_k < 1$ for all k. Let a be one of the decision surfaces and $m_1(0) < a$ and $m_2 > a$. Without loss of generality, we can assume $a = 0$, then $x \in c_1 \Longleftrightarrow x < 0$, $x \in c_2 \Longleftrightarrow x \geq 0$. Set

$$\theta_k := \frac{m_1(k) + m_2}{2}$$

then

$$x \in c_1(k) \Longleftrightarrow x < \frac{m_1(k) + m_2}{2} = \theta_k$$

and

$$x \in c_2(k) \Longleftrightarrow x \geq \frac{m_1(k) + m_2}{2} = \theta_k,$$

and Eq.(3) becomes

$$\theta_{k+1} = \theta_k + \alpha_{k+1} N_k \left(\frac{x(k) + m_2}{2} - \theta_k\right) \tag{4}$$

where

$$N_k = \begin{cases} -1 & \text{if } 0 \leq x(k) \leq \theta_k, \ \theta_k \neq 0 \\ 1 & \text{if } 0 > x(k) \geq \theta_k \\ 0, & \text{otherwise} \end{cases}$$

and m_2 is a fixed constant.

The following properties are held for $k \geq 1$.

(i) $m_1(k) \leq 0$.

(ii) $\theta_{k+1} > \theta_k \Rightarrow \theta_k < 0$ and $\theta_{k+1} < \theta_k \Rightarrow \theta_k > 0$.

(iii) $\theta_k < 0 \Rightarrow \theta_{k+1} \geq \theta_k$ and $\theta_k > 0 \Rightarrow \theta_{k+1} \leq \theta_k$.

(iv) $K_2 > m_2 \geq \theta_k > m_1(k) \geq -K_2$, where $K_2 = m_2 + |m_1(0)|$

The proof of these properties is in [3]:

Note that the proof of (ii) and (iii) shows us

$$(x(k) - m_1(k)) \geq 0, \quad if \quad N_k \neq 0 \tag{5}$$

and from (iv) and the definition of N_k, it follows that

$$|N_k(x(k) - m_1(k))| \leq |N_k x(k)| + |m_1(k)| \leq |\theta_k| + |m_1(k)| \leq 2K_2 \tag{6}$$

To prove the theorem, it suffices to prove that either

(A) algorithm of Eq.(3) stops at θ_{n_0} for some finite n_0; i.e. $\theta_k = \theta_{n_0}$ for $k \geq n_0$, or

(B) $\theta_k \to 0$ as $k \to +\infty$

Suppose that (A) is not true, then $\theta_n \neq 0$ for all n, otherwise if $\theta_{n_0} = 0$ for some n_0, then $N_{n_0} = 0$ and $\theta_{n_0+1} = 0, \cdots$, hence $\theta_n = \theta_{n_0} = 0$ for $n \geq n_0$.

Set

$$T_0 = 0, \ T_{n+1} = \inf\{k > T_n : \theta_{T_n}\theta_k < 0\} \ \text{ or } \ T_{n+1} = +\infty, \ \text{ if } \{\cdot\} \text{ is empty set.}$$

Note that N_k has the same sign for $k = T_n, \ T_n + 1, \ \cdots, \ T_{n+1} - 1$, and $\theta_{T_n}, \cdots, \theta_{T_{n+1}}$ is monotonous. Therefore from Eq.(4) and (5) and condition (iv), it follows that

$$2K_2 \geq |\theta_{T_{n+1}} - \theta_{T_n}| \geq \sum_{i=T_n}^{T_{n+1}-1} \alpha_{i+1}(x(i) - m_1(i))$$

From properties (ii) and (iii), it follows that if $\theta_{T_n} < 0$, then

$$\theta_{T_n} \leq \theta_{T_n+1} \leq \cdots \leq \theta_{T_{n+1}-1} < 0 \leq \theta_{T_{n+1}}$$

and if $\theta_{T_n} > 0$, then

$$\theta_{T_n} \geq \theta_{T_n+1} \geq \cdots \geq \theta_{T_{n+1}-1} > 0 \geq \theta_{T_{n+1}}.$$

If $T_{n+1} = +\infty$, then

$$2K_2 \geq \sum_{i=T_n}^{\infty} \alpha_{i=1}(x(i) - m_1(i))$$

and From Eq.(4),

$$\theta_t = \theta_{T_n} + N \sum_{i=T_n}^{t-1} \alpha_{i+1}(x(i) - m_1(i)),$$

where $N = 1$ or -1 is a constant. Thus, θ_t converges, and $(x(k) - m_1(k)) \to 0$ because of $\sum_{i=1}^{\infty} \alpha_{i+1} = +\infty$. If $N = 1$, and $0 > x(k) \geq \theta_k$, then $x(k) - m_1(k) \geq \frac{m_2 - m_1(k)}{2} \geq \frac{m_2}{2}$, hence $0 \geq \frac{m_2}{2}$. This is impossible. If $N = -1$, and $0 \leq x(k) \leq \theta_k$, then θ_k is decreasing, so is $m_1(k)$. But $m_1(k) \leq 0 \leq x(k)$. This contradicts to the fact $(x(k) - m_1(k)) \to 0$. Therefore, $T_{n+1} < +\infty$ for all n. Taking Eq.(4) and (6) into account, we obtain

$$|\theta_{T_{n+1}}| \leq |\theta_{T_{n+1}} - \theta_{T_{n+1}-1}| = \alpha_{T_{n+1}}(\frac{x(T_{n+1} - 1) + m_2}{2} - \theta_{T_{n+1}-1}) \leq 2K_2\alpha_{T_{n+1}} \to 0 \ \ as \ n \to +\infty$$

Finally, since each θ_k lies in between θ_{T_n} and $\theta_{T_{n+1}}$ for some n. We conclude that $\theta_k \to 0$ as $k \to +\infty$. Hence $m_1(k)$ has a limit as $k \to +\infty$.

Corollary 1 *If there exist two decision surfaces $a_1 \neq a_2$, then case (A) must happen. In the special case of a finite number of training patterns the algorithm will stop after some finite steps.*

Note that the above proof holds for both a_1 and a_2. If (A) doesn't happen, then $\theta_k \to a_1$ and $\theta \to a_2$ simultaneously, which is impossible. Therefore (A) must happen.

In order to prove the two-dimensional case, we need the following two assumptions:

Assumption I: Classes c_1 and c_2 are strictly linear separable; i.e. there exist two parallel straight lines such that class c_1 falls on one side of these two lines and class c_2 falls on the other side.

Assumption II: Classes c_1 and c_2 are bounded in a square region Λ : $[-M, \ M] \times [-M, \ M]$, where $M > 0$.

Note that under Assumption I there exists at least one decision surface. Hence the y-axis can be chosen to be the one among them such that for each training element $\mathbf{x} = (x_1, \ x_2)$,

$$x_1 > 0 \ \text{ or } \ x_1 < -\epsilon_0 \text{ for some fixed positive } \epsilon_0$$

and the x-axis can be chosen passing the point $\mathbf{m}_2 = (m_{21}, 0)$ where $M > m_{21} > 0$, which is the fixed prototype vector for class c_2. The initial value $\mathbf{m}_1(0)$ of the prototype vector for class c_1 is always chosen with properties $-M \leq m_{11}(0) \leq -\epsilon_0$ and $|m_{12}(0)| \leq M$. Then it is easy to see that

$$\mathbf{x} \in c_1 \Longleftrightarrow x_1 < 0, \ \mathbf{x} \in c_2 \Longleftrightarrow x_1 \geq 0$$

$$\mathbf{x} \in c_1 \Longleftrightarrow \Delta(k) := (\mathbf{m}_2 - \mathbf{m}_1(k))^T(\mathbf{x} - \frac{\mathbf{m}_2 + \mathbf{m}_1(k)}{2}) < 0$$

$$\mathbf{x} \in c_2(k) \Longleftrightarrow \Delta(k) \geq 0$$

and

$$N_k = +1 \Longleftrightarrow x_1 < 0 \quad \text{and} \quad \Delta(k) \geq 0, \tag{7}$$

$$N_k = -1 \Longleftrightarrow x_1 > 0 \quad \text{and} \quad \Delta(k) < 0. \tag{8}$$

Lemma 1 *Under Assumption I, $m_{11}(k) \leq -\epsilon_0$, for all k.*

Lemma 2 *Under Assumption II, $|m_{1i}(k)| \leq 10M$, for all k, $i = 1, \ 2$.*

Due to Lemmas 1 and 2, our algorithm becomes

$$\mathbf{m}_1(k+1) = \mathbf{m}_1(k) + \alpha_{k+1} N_k(\mathbf{x}(k) - \mathbf{m}_1(k))$$

$\mathbf{m}_1(k) \in D := \{(\omega_1, \ \omega_2) : \ -10M \leq \omega \leq -\epsilon_0, \ |\omega_2| \leq 10M\}$. Now we are going to prove the following theorem

Theorem 2 *Under Assumption I and II, Algorithm (3) either*
(i) stops at $m(k_0)$ for some finite k_0, i.e. $\mathbf{m}_1(k) = \mathbf{m}_1(k_0), \ k \geq k_0$, or
(ii) converges to some point.

Proof: Suppose that (i) is not true, then $N_k \neq 0$ for an infinite number of k. Without loss of generality, we assume that $N_k \neq 0$ for all k, hence from (7) and (8), it follows that

$$P(\{x_1(k) \geq 0, \Delta(k) < 0\} \cup \{x_1(k) < 0, \Delta(k) \geq 0\}) > 0 \quad \forall k \tag{9}$$

Considering the inputs $\mathbf{x}(k), \ k = 0, \ 1, \ \cdots$ are i.i.d random variables. Algorithm (3) is the so-called Robbins-Monro with

$$H(\mathbf{m}_1(k), \mathbf{x}(k)) = N_k(\mathbf{x}(k) - \mathbf{m}_1(k)).$$

Since for any positive Borel function g

$$E[g(\mathbf{m}_1(k), \mathbf{x}(k))|\mathcal{F}_k] = \int g(\mathbf{m}_1(k), \mathbf{x}) \mu_{\mathbf{m}_1(k)}(d\mathbf{x})$$

where $\mu_\omega(d\mathbf{x}) = p(x_1, x_2)d\mathbf{x}$ is independent of ω, $p(x_1, x_2)$ is the mixed density of classes c_1 and c_2 and $\mathcal{F}_k = \sigma\{\mathbf{m}_1(0), \ \cdots, \ \mathbf{m}_1(k), \mathbf{x}(0), \ \cdots, \ \mathbf{x}(k-1)\}$[4].

We introduce the so-called Lyapunov function as follows:

$$U(\omega) = \frac{\omega_2^2 + (\omega_1 + m_{21})^2}{-2\omega_1} \geq 0, \ \omega \in D$$

$$U(\omega) = 0 \Leftrightarrow \omega = \omega^* := (-m_{21}, 0)$$

$$U'(\omega) = \begin{pmatrix} -\frac{1}{2} + \frac{\omega_2^2 + m_{21}^2}{2\omega_1^2} \\ -\frac{\omega_2}{\omega_1} \end{pmatrix}$$

$$U''(\omega) = \begin{pmatrix} -\frac{\omega_2^2 + m_{21}^2}{\omega_1^3} & \frac{\omega_2}{\omega_1^2} \\ \frac{\omega_2}{\omega_1^2} & -\frac{1}{\omega_1} \end{pmatrix}$$

Using a Taylor expansion of order 2, we obtain:

$$U(\mathbf{m}_1(k+1)) = U(\mathbf{m}_1(k)) + \alpha_{k+1}[U'(\mathbf{m}_1(k))]^T H(\mathbf{m}_1(k), \mathbf{x}(k)) + r_k \tag{10}$$

where

$$r_k = \alpha_{k+1}^2 H^T(\mathbf{m}_1(k), \mathbf{x}(k)) U''(\bar{\omega}) H(\mathbf{m}_1(k), \mathbf{x}(k)), \quad \bar{\omega} \in [\mathbf{m}_1(k), \mathbf{m}_1(k+1)]$$

First, we verify the condition

$$\sup_{\omega \in D(\epsilon)} U'(\omega)^T h(\omega) < 0 \tag{11}$$

where

$$D(\epsilon) := \{\omega : \epsilon \le |\omega - \omega^*| \le \frac{1}{\epsilon}, \omega \in D\} \quad \text{for} \quad \epsilon > 0,$$

and

$$h(\omega) = E[H(\omega, \mathbf{x})] = \int H(\omega, \mathbf{x}) \mu_\omega(d\mathbf{x})$$

Note that

$$\begin{aligned} E[H(\omega, \mathbf{x})] &= \int\int N_k(\omega, \mathbf{x})(\mathbf{x} - \omega) p(x_1, x_2) dx_1 dx_2 \\ &= \int_{x_1 < 0} dx_1 \int_{\Delta \ge 0} (\mathbf{x} - \omega) p(x_1, x_2) dx_2 \\ &+ \int_{x_1 > 0} dx_1 \int_{\Delta < 0} (\omega - \mathbf{x}) p(x_1, x_2) dx_2 \end{aligned}$$

where $\Delta = (\mathbf{m}_2 - \omega)^T (\mathbf{x} - \frac{\mathbf{m}_2 + \omega}{2})$. We need to prove

$$U'(\omega)^T (\mathbf{x} - \omega) < 0 \ \ as \ \ x_1 < 0 \ \ \Delta \ge 0$$

$$U'(\omega)^T (\omega - \mathbf{x})) < 0 \ \ as \ \ x_1 > 0 \ \ \Delta < 0$$

If $x_1 > 0$, $\Delta = (m_{21} - \omega_1)x_1 - \omega_2 x_2 - \frac{m_{21}^2 - \omega_1^2 - \omega_2^2}{2} < 0$, then

$$\begin{aligned} U'(\omega)^T (\omega - \mathbf{x}) &= (-\frac{1}{2} + \frac{\omega_2^2 + m_{21}^2}{2\omega_1^2})(\omega_1 - x_1) + \frac{\omega_2}{\omega_1}(x_2 - \omega_2) \\ &= -\frac{\omega_2^2 + (m_{21} - \omega_1)^2}{2\omega_1^2} x_1 - \frac{\Delta}{\omega_1} < 0 \quad (\text{since } \omega_1 < 0) \end{aligned}$$

Similarly, if $x_1 < 0$, $\Delta > 0$, we obtain

$$U'(\omega)^T (\mathbf{x} - \omega) = \frac{\omega_2^2 + (m_{21} - \omega_1)^2}{2\omega_1^2} x_1 + \frac{\Delta}{\omega_1} < 0$$

Hence (11) holds because of (9).

Next we will verify the condition

$$E[||r_k||\mathcal{F}_k] \leq C\alpha_{k+1}^2(1 + U(\mathbf{m}_1(k))) \tag{12}$$

The left hand side of (12) equals to

$$\begin{aligned} E[r_k||\mathcal{F}_k] &= \alpha_{k+1}^2 E[|N_k^2(\mathbf{x}(k) - \mathbf{m}_1(k))^T U''(\bar{\omega})(\mathbf{x}(k) - \mathbf{m}_1(k))||\mathcal{F}_k] \\ &\leq C\alpha_{k+1}^2 \end{aligned}$$

for some positive C, since $\bar{\omega} \in D$, $\bar{\omega_1}$, $m_{11}(k)$ are bounded from 0 and $\mathbf{m}_1(k)$, $\mathbf{x}(k)$ are bounded. By the Assumptions I, II and lemma 1 and 2, we obtain

$$E[||r_k||\mathcal{F}_k] \leq \alpha_{k+1}^2 C(1 + U(\mathbf{m}_1(k)))$$

Now from (10), (11) and (12), $U(\mathbf{m}_1(k))$, $k = 1, 2, \cdots$, satisfies

$$E[U(\mathbf{m}_1(k+1))|\mathcal{F}_k] = (1 + \alpha_{k+1}^2 C)U(\mathbf{m}_1(k)) + C\alpha_{k+1}^2 + U'(\mathbf{m}_1(k))^T h(\mathbf{m}_1(k))$$

By the same discussion in p.345 section 5.2.2 in [4], we conclude that algorithm (3) converges to ω^*.

Corollary 2 *Under Assumptions I and II, algorithm (3) must stop at some finite steps.*

Corollary 3 *If there are a finite number of training elements, then algorithm (3) must stop at some finite step.*

3 Conclusion

In this paper, the convergence of the learning rules for pattern classification under some conditions is analyzed for a finite number of training samples to a fixed point. This has also provided the mathematical basis for the analysis of the Kohonen learning vector quantization algorithms.

References

[1] Z.-P. Lo, Y. Q. Yu, and B. Bavarian , "Derivation of Learning Vector Quantization Algorithms," Submitted to *IJCNN,* 1992.

[2] T. Kohonen, G. Barna and R. Chrisley, "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies," *Proc. of the IJCNN,* vol.1 pp. 182-185 July, 1988.

[3] Z.-P. Lo, Y. Q. Yu, and B. Bavarian, "Analysis of a Learning Algorithm for Neural Network Classifiers," Submitted to *IEEE Trans. on Neural Networks,* 1992.

[4] A. Benveniste, M. Melivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations,* Springer-Verlag, Berlin Heidelberg, New York, 1990.

IMPROVING THE PERFORMANCE OF PROBABILISTIC NEURAL NETWORKS

M. T. Musavi, K. Kalantri, and W. Ahmed
Electrical and Computer Engineering
University of Maine
Orono, Maine 04469
musavi@watson.eece.maine.edu

Abstract

This paper presents a methodology for selection of appropriate widths or covariance matrices of the Gaussian functions in implementation of PNN classifiers. The Gram-Schmidt orthogonalization process is employed to find these matrices. It has been shown that the proposed technique improves the generalization ability of the PNN classifiers over the standard approach. The result can be applied to other Gaussian based classifiers such as the radial basis functions (RBF).

1. Introduction

The probabilistic neural network (PNN) classifiers have proven to be more time efficient than the conventional back-propagation (BP) type networks. The main idea is to use the Baysian decision rule to separate decision regions in a multi-dimensional input space. The rule is based on the minimization of the "expected risk" of misclassification. Specht in his earlier work (Specht, 1967) used a kernel estimator to estimate the input density function of the problem under consideration. He then used this estimator and the Bayesian rule to design a classifier that was later named probabilistic neural network (PNN) (Specht, 1990a; Specht 1990b). The significant advantage of the PNN classifier is its speed and training process. First, the training process is one-pass and without any iteration for weight adaptation, hence, yielding great processing speed as compared to back propagation (BP) or similar techniques (Specht & Shapiro, 1991). Second, the network generalizes to the new incoming patterns without having to repeat the training process. These characteristics are ideal for real time applications (Maloney, 1988; Maloney & Specht, 1989; Washburne , Okamura, Specht & Fisher, 1991).

An important issue that has not been given enough attention is the selection of an optimal "smoothing parameter" in the PNN classifiers. Smoothing parameter is the width or the covariance matrix of the Gaussian kernel function in the PNN classifier. The PNN decision boundary varies from a hyperplane to a very nonlinear boundary when the smoothing parameter varies from 0 to ∞. Specht indicates that the classification rates is not sensitive to this parameter. In our experiments with the PNN it has been observed that the choice of smoothing parameter can, in fact, have significant effect on the network outcome. This is especially evident in the statistical problems with relatively close patterns of opposite class. More over, it has been also experienced that the

0-7803-0559-0/92 $3.00 © 1992 IEEE

PNN will computationally fail to yield a solution when a small smoothing parameter is selected.

In view of the former deductions we offer an approach to improve the generalization ability of PNN classifiers by selecting many covariance matrices, with ellipsoid constant potential surface (CPS), as opposed to only one. The weighted probabilistic neural network (WPNN) (Montana, 1991) also offers a similar treatment of the issue but it falls short in providing a reliable approach for finding the eigenvalues of the covariance matrices. In our approach, however, the eigenvalues are found by the Gram-Schmidt orthogonalization process. The approach has been tested with different problems and improvements have been reported.

2. The Proposed Methodology

The underlying theory in the PNN classifier is to estimate the density of the input space from the given observations by linear combination of a set of local functions. For example in a two class problem the densities are found by:

$$f_A(\underline{x}) = \frac{1}{T} \sum_{i=1}^{T} \Phi_i (\|\underline{x}-\underline{c}^{ai}\|) \tag{1a}$$

$$f_B(\underline{x}) = \frac{1}{T} \sum_{i=1}^{T} \Phi_i (\|\underline{x}-\underline{c}^{bi}\|) \tag{1b}$$

where $\underline{c}^{ai}$ and $\underline{c}^{bi}$ are the training data points from class A and B respectively, $\Phi_i(.)$ is the kernel function at the i-th node and T is the total number of training patterns. The training in PNN is therefore that of finding a set of appropriate kernel functions and their parameters. The classification is based on the Bayes rule. To simplify the matters Specht has made the following three assumptions

i) All kernels are the same

$$\Phi_i (\|\underline{x}-\underline{c}^{ai}\|) = \Phi_i (\|\underline{x}-\underline{c}^{bi}\|) = \Phi (\|\underline{x}-\underline{c}^{i}\|). \qquad \text{for} \quad i=1, 2, ..., T \tag{2a}$$

ii) The kernel is Gaussian,

$$\Phi (\|\underline{x}-\underline{c}^{i}\|) = (2\pi)^{-M/2} |\Sigma|^{-1/2} \exp\left[-\frac{1}{2}(\underline{x} - \underline{c}^{i})^T [\Sigma]^{-1} (\underline{x} - \underline{c}^{i}) \right]. \tag{2b}$$

M is dimension of the input space and Σ is the covariance matrix.

iii) And that the covariance matrix Σ is diagonal and has equal eigenvalues ($\Sigma=\sigma I$),

$$\Phi (\|\underline{x}-\underline{c}^{i}\|) = (2\pi)^{-M/2} \sigma^{-M} \exp\left[-\frac{\|\underline{x} - \underline{c}^{i}\|^2}{\sigma^2} \right] \tag{2c}$$

Using the above assumptions the problem of finding kernel functions is simply reduced to selecting the parameter σ that is referred to as smoothing parameter. This parameter is normally found by try and error. It is necessary that the kernel functions preserve the local properties of the

input space. The above assumptions will not satisfy this condition. Therefore, the PNN classifier will not be able to achieve its optimal performance.

So, let's start with our technique and consider the general case and disregard all the above-mentioned assumptions. In that, the kernels are still Gaussian but every one of them can take a different covariance matrix. The Gaussian kernel placed at $\underline{c}^i$ is then indicated by:

$$\Phi_i (\|\underline{x}-\underline{c}^i\|) = (2\pi)^{-M/2} |\Sigma_i|^{-1/2} \exp\left[-\frac{1}{2}(\underline{x}-\underline{c}^i)^T [\Sigma_i]^{-1} (\underline{x}-\underline{c}^i)\right]. \quad (3)$$

Our goal is to estimate the covariance matrix in such a way to minimize the overlapping of the nearest neighbors of different classes to preserve local properties, as well as maximize the generalization ability of the network. In general, the contour or the constant potential surface (CPS) of the Gaussian function is an ellipsoid in a multi-dimensional space. The actual CPS is controlled by Σ_i. Therefore, the generalization of the classifier is determined by Σ_i. In order to obtain good generalization, the eigenvalues of the covariance matrix of each Gaussian should be as large as possible. While there is no problem if two functions belonging to the same class overlap each other, functions of different classes have to be separated to avoid any significant overlapping. So, the function is constrained by the locations of the nearest training patterns of the other class.

Now, let's decompose the i-th covariance matrix as,

$$\Sigma_i = Q_i \Lambda_i Q_i^T \quad (4)$$

where eigenvalues and eigenvectors are respectively the diagonal entries of Λ_i and the columns of Q_i. Since Σ_i is symmetric, all eigenvalues and their corresponding eigenvectors are real. The eigenvectors are the principal axes of the ellipsoid of the CPS and the square roots of eigenvalues define the lengths of the ellipsoid along these principal axes. To find the principal axes and the lengths along the axes for the CPS ellipsoid the Gram-Schmidt orthogonalization procedure is utilized. The j-th eigenvalue of the i-th covariance matrix can be found by (Musavi, 1991):

$$\lambda_{ij} = \frac{\|\underline{b}_{ij}\|^2}{4r^2}. \quad (5)$$

Where $\|\underline{b}_{ij}\|$ is the length of the j-th axis of the i-th CPS ellipsoid and r is a measure related to the overlapping of the local functions. Once eigenvalues and eigenvectors are known the kernels can be found.

3. Test Results

To show the effect of the proposed technique on the generalization ability of the PNN classifiers we have conducted different tests. These tests are separated into two different

categories. In the first category the standard approach is used and in the second category our proposed approach is applied.

Two test problems with two (2-d) and eight (8-d) input dimensions have been generated by Gaussian random vectors. Without loss of generality it has been assumed that there are only two classes in each problem. In the 2-d problem the first class has zero mean random vectors with identity covariance matrix and the second class has mean vector [1 2] and diagonal covariance matrix with 0.01 and 4.0 entries. The distribution of training samples for the 2-d problem is given in Figure 1. The first class patterns have been indicated by "x" and the second class by dots ".". The patterns of the 8-d problem are Gaussian random vectors with equal mean vectors and identity (I) and four-identity (4I) covariance matrices.

Note that for any Gaussian distribution the optimal classifier is known to be a quadratic classifier given by:

$$\frac{1}{2}(\underline{x}-\underline{m}_1)^T[\Sigma_1]^{-1}(\underline{x}-\underline{m}_1)-\frac{1}{2}(\underline{x}-\underline{m}_2)^T[\Sigma_2]^{-1}(\underline{x}-\underline{m}_2)+\frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|}\underset{\text{class2}}{\overset{\text{class1}}{\lesseqgtr}}0 \qquad (6)$$

where $\underline{m}_i$ is the mean vector of class i and Σ_i is the covariance matrix of class i. Applying (6) the optimal error rates for 10,000 samples of 2-d and 8-d problems were found to be 6% and 9% respectively. These optimal values will be compared with the results of our experiments.

For the first category of tests we selected 100, 150, 200, 250, and 300 training patterns along with 10,000 test patterns of each problem. As described by (2), the standard practice is to select σ by try and error. Therefore, we allowed σ to vary within an specified range. For any given number of training patterns we then trained the networks for different σ's and tested the networks with the given 10,000 test patterns. Figures 2a-b indicate the error rates of the PNN network for the 2-d and 8-d problems using the standard approach. Note that the error rate is sensitive to variations in σ, especially in the 2-d problem.

In the second category we used the proposed methodology to find the covariance matrices of the Gaussian functions as opposed to the standard approach. The best error rates of the standard approach were selected from Figures 2a-b and plotted against those of our approach. The results are shown in Figures 3a-b. Note that in all cases the proposed methodology for selecting appropriate covariance matrices gives better error rates than the standard approach. This is especially evident in the 8-d problem. The generalization of the enhanced PNN trained with low number of training samples is as good as or better than that of the standard PNN with high number of samples. For example, in the 8-d problem the best error rate of standard PNN for 300 training points was 25.69% while that of enhanced PNN with only 100 training points was 20.39%.