Probabilistic Neural-Network Structure Determination for Pattern Classification

K. Z. Mao, K.-C. Tan, and W. Ser

***Abstract*—Network structure determination is an important issue in pattern classification based on a probabilistic neural network. In this study, a supervised network structure determination algorithm is proposed. The proposed algorithm consists of two parts and runs in an iterative way. The first part identifies an appropriate smoothing parameter using a genetic algorithm, while the second part determines suitable pattern layer neurons using a forward regression orthogonal algorithm. The proposed algorithm is capable of offering a fairly small network structure with satisfactory classification accuracy.**

***Index Terms*—Genetic algorithms, orthogonal algorithm, pattern classification, probabilistic neural network (PNN).**

I. Introduction

Applications of neural network to pattern classification have been extensively studied in the past many years. Various kinds of neural-network architecture including multilayer perceptron (MLP) neural network, radial basis function (RBF) neural network, self-organizing map (SOM) neural network, and probabilistic neural network (PNN) have been proposed. Because of ease of training and a sound statistical foundation in Bayesian estimation theory, PNN has become an effective tool for solving many classification problems (e.g., [7], [10], [11], [13], [16]). However, there is an outstanding issue associated with PNN concerning network structure determination, that is determining the network size, the locations of pattern layer neurons as well as the value of the smoothing parameter. As a matter of fact, the pattern layer of a PNN often consists of all training samples of which many could be redundant. Including redundant samples can potentially lead to a large network structure, which in turn induces two problems. First, it would result in higher computational overhead simply because the amount of computation necessary to classify an unknown pattern is proportional to the size of the network. Second, a consequence of a large network structure is that the classifier tends to be oversensitive to the training data and is likely to exhibit poor generalization capacities to the unseen data ([2]). On the other hand, the smoothing parameter also plays a crucial role in PNN classifier, and an appropriate smoothing parameter is often data dependent.

The two problems mentioned above have been realized by some researchers and some algorithms for reduction of training samples have been proposed ([3], [12], [14], [17], [21]). The vector quantization approach was employed to group training samples and find cluster centers to be used for PNN in [3] and [21]. In [12] and [17], the probability density function of a PNN was approximated by a small number of component densities and the parameters of the components were estimated from the training set by using a Gaussian clustering self-organizing algorithm. In [14], the clustering technique of the restricted Coulomb energy paradigm was used to find cluster centers and associated weights corresponding to the number of samples represented by each cluster. Basically, all the above mentioned PNN reduction algorithms are based on the clustering approach. Since the classification error has not been used directly in the process of neuron selection, these algorithms can be classified into the category of unsupervised learning.

In this study, we propose a supervised PNN structure determination algorithm. A strength of this supervised learning algorithm is that the requirements on classification error rate and model size are incorporated directly in the process of determining the network structure. Indeed, we propose to solve the PNN structure determination problem by minimizing the network size under the constraint of meeting a specific classification error rate. The proposed algorithm consists of two parts and runs in an iterative way. The first part of the algorithm performs smoothing parameter selection. Since there is no known quantitative relationship among the network size, classification error rate and the smoothing parameter, a genetic algorithm (GA), instead of others that demand such a quantitative relationship, is employed to find an appropriate smoothing parameter. The second part of the proposed algorithm performs pattern layer neuron selection. With the use of the smoothing parameter already determined in the first part, the output of a summation layer neuron becomes a linear combination of the outputs of pattern layer neurons. Subsequently, an orthogonal algorithm ([1], [4]) is employed to select important neurons. Because of the incorporation of an orthogonal transform in neuron selection, the proposed algorithm is computationally more efficient than other algorithms that use genetic algorithms (GAs) to search all parameters of the neural networks structure (e.g., [20]).

This paper is organized as follows. In Section II, a brief overview of the PNN is presented and the associated problems are analyzed. Pattern layer neuron selection using an orthogonal algorithm is discussed in Section III-A. In Section III-B, a PNN smoothing parameter selection algorithm is proposed. Numerical examples are presented in Section IV to demonstrate the effectiveness of the proposed algorithm.

II. A Brief Review of the PNN's

The PNN was first proposed in [13]. The architecture of a typical PNN is as shown in Fig. 1. The PNN architecture is composed of many interconnected processing units or neurons organized in successive layers. The input layer unit does not perform any computation and simply distributes the input to the neurons

Manuscript received June 3, 1999; revised January 20, 2000.

The authors are with the Centre for Signal Processing, Nanyang Technological University, Singapore.

Publisher Item Identifier S 1045-9227(00)05954-3.

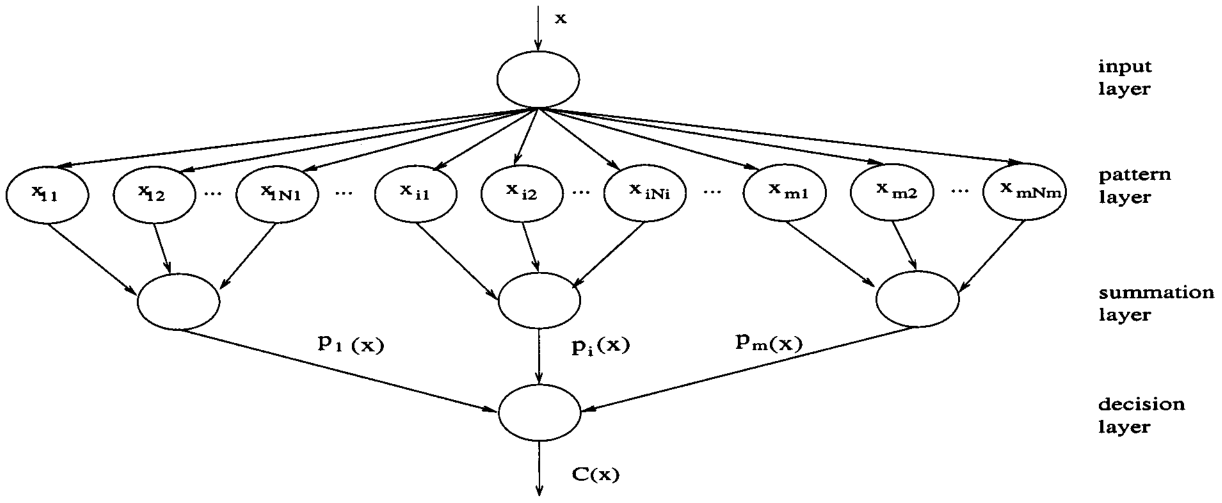

Fig. 1. Diagram of a PNN.

in the pattern layer. On receiving a pattern x from the input layer, the neuron x_{ij} of the pattern layer computes its output

$$\phi_{ij}(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left[-\frac{(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right] \tag{1}$$

where d denotes the dimension of the pattern vector x, σ is the smoothing parameter and x_{ij} is the neuron vector. The summation layer neurons compute the maximum likelihood of pattern x being classified into C_i by summarizing and averaging the output of all neurons that belong to the same class

$$p_i(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{N_i} \sum_{j=1}^{N_i} \exp\left[-\frac{(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right] \tag{2}$$

where N_i denotes the total number of samples in class C_i. If the *a priori* probabilities for each class are the same, and the losses associated with making an incorrect decision for each class are the same, the decision layer unit classifies the pattern x in accordance with the Bayes's decision rule based on the output of all the summation layer neurons

$$\hat{C}(x) = \arg\max\{p_i(x)\}, \qquad i = 1, 2, \cdots, m \tag{3}$$

where $\hat{C}(x)$ denotes the estimated class of the pattern x and m is the total number of classes in the training samples. One outstanding issue associated with the PNN is the determination of the network structure. This includes determining the network size, the pattern layer neurons and an appropriate smoothing parameter. Some algorithms for pattern layer neurons selection have been proposed ([3], [12], [14], [15], [17], [21]). Since the classification error has not been used directly in the process of PNN structure determination, the algorithms mentioned above can be classified into the category of unsupervised learning.

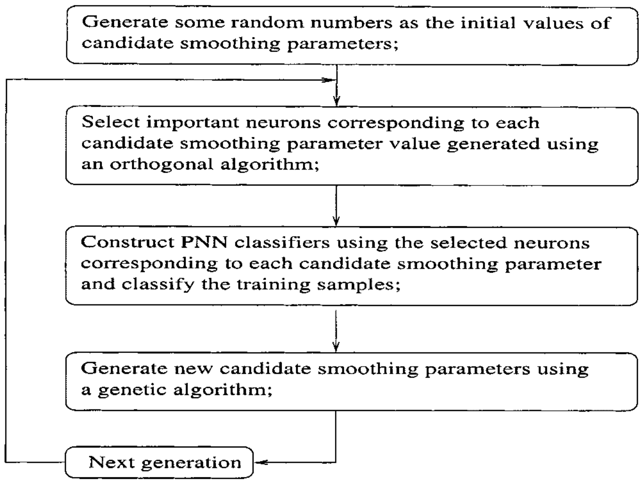
Generate some random numbers as the initial values of candidate smoothing parameters;
Select important neurons corresponding to each candidate smoothing parameter value generated using an orthogonal algorithm;
Construct PNN classifiers using the selected neurons corresponding to each candidate smoothing parameter and classify the training samples;
Generate new candidate smoothing parameters using a genetic algorithm;
Next generation

Fig. 2. Diagram of the proposed PNN structure determination algorithm.

III. Determining the PNN Structure Using an Orthogonal Algorithm and the Genetic Algorithm

In this section, we propose a supervised PNN structure determination algorithm that incorporates an appropriate constraint on classification error rate. The proposed algorithm consists of two parts and runs in an iterative way as shown in Fig. 2, of which the first part identifies a suitable smoothing parameter using a genetic algorithm (GA) [6], while the second part performs pattern layer neurons selection using an orthogonal algorithm [4]. Recently, an algorithm with similar architecture for RBF neural networks structure determination was proposed [5]. However, we developed our algorithm independently, and submitted our work for review before the publication of [5]. The difference bewteen our algorithm and that in [5] is discussed in Section III-A.

A. Construct the Pattern Layer Using the Forward Regression Orthogonal Algorithm

At this stage, it is assumed that the smoothing parameter has been chosen. The objective is to select representative pattern layer neurons from the training samples. As described in the previous section, for the kth training pattern in class C_i denoted by vector x_{ik}, the maximum likelihood to be classified to C_i is

$$\begin{aligned} p_i(x_{ik}) &= \frac{1}{(2\pi)^{d/2}\sigma^d}\,\frac{1}{N_i}\sum_{j=1}^{N_i} \\ &\quad \cdot \exp\left[-\frac{(x_{ik}-x_{ij})^T(x_{ik}-x_{ij})}{2\sigma^2}\right] \\ &= \sum_{j=1}^{N_i}\phi_{ij}(x_{ik}) \end{aligned} \tag{4}$$

where

$$\phi_{ij}(x_{ik}) = \frac{1}{(2\pi)^{d/2}\sigma^d}\,\frac{1}{N_i}\exp\left[-\frac{(x_{ik}-x_{ij})^T(x_{ik}-x_{ij})}{2\sigma^2}\right].$$

Note that $p_i(x_{ik})$ is a nonlinear function of the smoothing parameter and the pattern layer neuron vectors x_{ij}. But if the smoothing parameter is set to a prespecified value, and the output of each neuron $\phi_{ij}(x_{ik})$ is considered as an auxiliary regression variable, $p_i(x_{ik})$ becomes a linear combination of these auxiliary variables as shown in (4). Linear orthogonal transforms can therefore be applied to decompose the coupling between these auxiliary variables so as to facilitate an evaluation of the importance of each neuron.

Equation (4) can alse be written in a matrix form as

$$P = \Phi\theta \tag{5}$$

where

$$\begin{aligned} \theta &= [1, 1, \cdots, 1]^T \\ P &= [p_i(x_{i1}), p_i(x_{i2}), \cdots, p_i(x_{iN_i})]^T \\ \Phi &= \begin{bmatrix} \phi_{i1}(x_{i1}) & \phi_{i2}(x_{i1}) & \cdots & \phi_{iN_i}(x_{i1}) \\ \phi_{i1}(x_{i2}) & \phi_{i2}(x_{i2}) & \cdots & \phi_{iN_i}(x_{i2}) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{i1}(x_{iN_i}) & \phi_{i2}(x_{iN_i}) & \cdots & \phi_{iN_i}(x_{iN_i}) \end{bmatrix}. \end{aligned}$$

Applying an orthogonal transform to the regression matrix Φ, one can obtain

$$\Phi = QR = [Q_1, Q_2, \cdots, Q_{N_i}]R \tag{6}$$

where $Q_1, Q_2, \cdots, Q_{N_i}$ are an orthogonal basis, R is a triangular matrix as follows:

$$R = \begin{bmatrix} 1 & r_{12} & r_{13} & \cdots & r_{1N_i-1} & r_{1N_i} \\ 0 & 1 & r_{23} & \cdots & r_{2N_i-1} & r_{2N_i} \\ \vdots & \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \cdots & 1 & r_{N_i-1N_i} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

The importance of the jth candidate neuron in class C_i is evaluated based on the norm of Q_j

$$\Gamma_j = Q_j^T Q_j. \tag{7}$$

Under the condition that all neurons have the same smooth parameter, a larger value of Γ_j indicates that more neurons are closest to the corresponding neuron. Therefore the larger the Γ is, the more important the corresponding neuron is. The criterion given by (7) is simple to implement and often provides a satisfactory evaluation of the importance of the neurons.

The neuron importance evaluating and ranking procedure is summarized as follows:

1) To determine the most representative neuron for class C_i, all the N_i training samples in class C_i are considered as the possible candidate. It requires a computation of the output of each candidate neuron to classify all training samples in class C_i

$$\begin{aligned} Q_1^{(\alpha)} &= \phi_\alpha, \qquad \alpha = 1, 2, \cdots, N_i; \\ \phi_\alpha &= [\phi_\alpha(1), \phi_\alpha(2), \cdots, \phi_\alpha(N_i)]^T. \end{aligned} \tag{8}$$

The importance is evaluated as follows:

$$\Gamma_1^{(\alpha)} = [Q_1^{(\alpha)}]^T Q_1^{(\alpha)}, \qquad \alpha = 1, 2, \cdots, N_i. \tag{9}$$

The sample that corresponds to the maximum importance index, say x_{ij}, is considered as the most representative neuron for class C_i and is used to generate Q_1.

2) To determine the jth representative neuron for class C_i, all the remaining $N_i - j + 1$ samples in class C_i, say $\phi_{k_1}, \phi_{k_2}, \cdots, \phi_{k_{N_i-j+1}}$, are considered as the candidate. Compute the importance index

$$\begin{aligned} Q_j^{(\alpha)} &= \phi_{k_\alpha} - \sum_{l=1}^{j-1} r_{l\alpha}^{(\alpha)} Q_l, \\ &\qquad \alpha = 1, 2, \cdots, N_i - j + 1 \end{aligned} \tag{10}$$

$$\begin{aligned} r_{l\alpha}^{(\alpha)} &= Q_l^T \phi_{k_\alpha} / Q_l^T Q_l, \\ &\qquad \alpha = 1, 2, \cdots, N_i - j + 1, \quad l < i. \end{aligned} \tag{11}$$

The sample with the maximum importance index is considered as the jth representative neurons.

The orthogonal transform has been employed in determining the structures of RBF and fuzzy basis funcion (FBF) neural networks ([4], [5], [18], [19]). If extended to multiple-output systems, the algorithms in [4], [5] are applicable to pattern classification problems. Determining the structure of the mutiple-output RBF structure using the extended algorithms of [4], [5] is equlivalent to determining the structure of the PNN based on the minimization of the mean squared error (MSE) between the output vectors of summation layer neurons and the desired output vectors, whose elements can be set to one or zero to indicate the input patterns belonging to or not belonging to the class. However, for the PNN-based classifier, one is more concerned with the error of decision layer than that of the summation layer since a smaller MSE of the summation layer does not necessarily lead to a better classification. Therefore we have chosen to determine the structure of the PNN based on the minimization of the decision layer error, in which the Bayesian decision rule is incorporated in the process of determining network structure. This is the main difference between our algorithm and those in [4], [5]. In addition, algorithms developed in the present study is applicable to problems with small numbers of training samples. Indeed, PNN's were considered as variants of RBF neural networks. But unlike the RBF neural networks whose weights are adjustable, the weights of PNN's are all set to one. Probably because of this, PNN's is applicable to problems with a small number of training samples and is less likely to overfit the training data. This is an advantage of PNN since a small number of training sample is often encountered in pattern classification. For example, only seven training samples are available for each class in Experiment 3.

Based on the sample importance evaluating and ranking procedure Steps 1), 2), the PNN classifier construction procedure is summarized as follows.

1) Select the most representative neuron for each class from all the training samples using the neuron importance evaluating and ranking procedure Step 1).

2) Construct a probabilistic neural-network classifier using all the selected representative neurons. Classify the training samples in each class and compute the classification error rate, which is defined as the ratio of the number of misclassifications to the number of training samples in each class.

3) Select one additional representative neuron using the neuron importance evaluating and ranking procedure Step 2) for classes that the requirement on classification error rate is not satisfied.

4) Goto Step 2) until the requirement on classification error rate of all classes are satisfied. If the training samples are poor, the required classification accuracy might not be met even if all the training samples are used to construct the pattern layer. If this is the case, a higher classification error rate will be used.

Because only the most important neuron is selected at every step, the above procedure is capable of selecting a fairly small PNN with satisfactory classification accuracy.

B. Selecting the Smoothing Parameter Using Genetic Algorithms

In Section III-A, it is assumed that the smoothing parameter is set to a prespecified value. But an appropriate smoothing parameter is often data dependent. Therefore, it requires a proper procedure for smoothing parameter selection as we shall propose here.

When a neural-network classifier is constructed, classification accuracy and network size are the most important aspects that need to be taken into consideration. Often it is desired that the network architecture could be minimized under the condition that the classification error rate is smaller than a prespecified tolerance bound. Therefore we solve the PNN structure determination as the following constrained optimization problem in the present study

$$\min\{n\} \tag{12}$$

subject to

$$\mu < \delta \tag{13}$$

where n denotes the network size, i.e., the number of selected pattern layer neurons, μ is the classification error rate which is defined as the ratio between the number of misclassifications and all the training samples, δ is a prespecified upper bound of classification error tolerance. Since there is no known quantative relationship among the network size, the classification error rate and the smoothing parameter, a GA [6], instead of others that demand such a quantative relationship, is empolyed to slove the above optimization problem. The GA-based network structure determination algorithm is outlined as follows.

1) Generate a set of random real number for the smoothing parameter σ.
2) Set the smoothing parameter to the values already defined, construct classifiers using the orthogonal algorithm-based PNN construction procedure Steps 1)–4) of Section III-A.
3) Classify all the training samples and compute the classification error rate for each class.
4) Perform genetic operations on the values of smoothing parameter and generate a new set of smoothing parameter.
5) Go to Step 2) until the number of iteration reaches a prespecified value.

GAs are a class of random search procedure which were initially motivated by the principles of natural evolution and population genetics ([6]). GA-based optimization is a guided random search method which could find an optimal solution without exaustively testing all possible solutions.

Typically, a genetic algorithm consists of the following operations; encoding, fitness value assignment, reproduction, crossover and mutation. Details of the GA-based PNN structure detection algorithm are described below.

1) Encoding: GA works with the coding of parameters rather than the parameter themself. If samples are normalized, the smoothing parameter should be smaller than one, only fraction part needs to be coded. A four-bit decimal coding is employed in the present study to encode the smoothing parameter σ. For example, one individual is $\sigma = 0.6257$, this value can be represented by the following decimal string:

$$\overbrace{6}^{b_1}\quad\overbrace{2}^{b_2}\quad\overbrace{5}^{b_3}\quad\overbrace{7}^{b_4}$$

where b_i denote the bit of 10^{-i}. The physical interpretation of the above string is that the 10^{-1} bit is six, the 10^{-2} bit is two, the 10^{-3} bit is five, the 10^{-4} bit is seven.

2) Fitness Evaluation: Each individual represents a smoothing parameter value. With the use of neuron selection algorithm developed in Section III-A and smoothing parameters defined by all individuals, a number of candidate network structures can be obtained. The objective is to minimize the neural-network size, therefore the fitness function should be inversely proportional to the number of selected neurons. The fitness can be computed using the following mapping scheme:

$$\rho_i = \rho_{\max} - \frac{\rho_{\max} - \rho_{\min}}{n_{\max} - n_{\min}}\,(n_i - n_{\min}) \tag{14}$$

where ρ_i denotes the fitness value of the ith individual. $n_{\min}$, $n_{\max}$, and $\rho_{\min}$, $\rho_{\max}$ are the minimum and maximum size of candidate network structure in the current population, and the minimum and maximum fitness values, respectively. In this study $\rho_{\min}$ and $\rho_{\max}$ are set to 0.5 and 1, respectively.

3) Reproduction: The *roulette wheel* approach is employed to implement the reproduction procedure. Each string is allocated a slot of the *roulette wheel* subtending an angle proportional to its fitness. A random number in the range of 0 to 2π is generated. A copy of string goes to the mating pool if the random number falls in the slot corresponding to the string. The reproduction is repeated to generate a mating pool with a prespecified size.

4) Crossover: The purpose of crossover operation is to generate new solutions by exchanging bits between individuals. Assuming two randomly selected parent individuals are given by string i and string j

$$\overbrace{6}^{b_1}\quad\overbrace{2}^{b_2}\quad\overbrace{5}^{b_3}\quad\overbrace{7}^{b_4}\qquad\text{string } i$$
$$\overbrace{0}^{b_1}\quad\overbrace{4}^{b_2}\quad\overbrace{3}^{b_3}\quad\overbrace{2}^{b_4}\qquad\text{string } j.$$

First, randomly select the bit at which the two strings will be changed, for example b_2. Then exchanging the values at bit b_2 for the two strings yields two offspring strings:

$$\overbrace{6}^{b_1}\quad\overbrace{4}^{b_2}\quad\overbrace{5}^{b_3}\quad\overbrace{7}^{b_4}\qquad\text{string } k$$
$$\overbrace{0}^{b_1}\quad\overbrace{2}^{b_2}\quad\overbrace{3}^{b_3}\quad\overbrace{2}^{b_4}\qquad\text{string } l.$$

5) Mutation: The purpose of employing mutation is to generate an individual that is not easy to achieve by the crossover operation. In this study, the mutation is achieved by changing the selected bit with a random number between zero to nine. For example if the bit b_1 of string k is supposed to mutate, changing this bit to a random generated number, say eight, yields the following string:

$$\overbrace{8}^{b_1}\quad\overbrace{4}^{b_2}\quad\overbrace{5}^{b_3}\quad\overbrace{7}^{b_4}\qquad\text{string } z.$$

6) Summary of the Proposed Algorithm: The proposed PNN network structure determination algorithm can be summarized as follows.

1) Generate an initial population set $\mathcal{P}$ consisting of n individuals, each individual represents a smoothing parameter. Set the current generation number $i = 1$.
2) With the use of neuron selection algorithm developed in Section III-A and smoothing parameters defined by all individuals, a number of candidate network structures can be obtained. Compute the fitness value. Form a mating pool $\mathcal{M}$ using all individuals in the population set $\mathcal{P}$ at the probabilities assigned to each individual proportional to the corresponding fitness value.
3) Randomly select a pair of parent strings from the mating pool $\mathcal{M}$. Choose a random crossover point and exchange the parent string bits to produce two offsprings and put the offsprings in the offspring set $\mathcal{O}$. The procedure is repeated until the number of offspring strings are the same as the number of parent strings.
4) Mutate each bit of each offspring in the set $\mathcal{O}$ with a prespecified mutation rate and calculate the fitness value of each mutated offspring using the procedure summarized in Step 2).
5) Select the n fittest individuals from sets $\mathcal{P}$ and $\mathcal{O}$ by comparing fitness values.
6) Reset the set $\mathcal{P}$ with the newly selected n individuals, reset the number of generations $i = i + 1$, and nullify the offspring set $\mathcal{O}$.
7) Steps 2)–6) are repeated until a prespecified number of generations arrives.

Remark 1: A single smoothing parameter is used in the algorithm developed above. This is the same as the original PNN. But employing a single smoothing parameter might not be a good choice in some cases. Consider a two-class problem, where samples in class A scatter widely, while samples in class B concentrate, and region A is very close to region B. Classifiers using a large smoothing parameter cannot capture the classes well. Employing a small smoothing parameter can capture the classes, but this would lead to a large network structure. To alleviate this problem, we can employ multiple smoothing parameters, where each class has a smoothing parameter. By modifying the coding scheme, the algorithm developed above is applicable to the case of multiple smoothing parameters. For example, for a two-class problem, $\sigma_1 = 0.7251$, $\sigma_2 = 0.2103$, the smoothing parameters are coded as follows:

$$\underbrace{\overbrace{7}^{b_1}\quad\overbrace{2}^{b_2}\quad\overbrace{5}^{b_3}\quad\overbrace{1}^{b_4}}_{\sigma_1}\quad\underbrace{\overbrace{2}^{b_5}\quad\overbrace{1}^{b_6}\quad\overbrace{0}^{b_7}\quad\overbrace{3}^{b_8}}_{\sigma_2}.$$

Remark 2: GAs have been used in neural-networks structure determination ([9], [20] and the references therein), in which all parameters of the networks were optimized using GAs. The parameters can be weights of the MLP neural networks, or locations of hidden layer neurons and width of RBFs of the RBF neural networks. In theory, the GAs could find the optimal solution. In practice, however, the optimal solution, even the suboptimal solution is difficult to find if the number of parameters to be optimized is very large. This is because the search space expands dramatically with the increase of the number of parameters. In our study, the GAs were used in a different way, where the pattern layer neurons were selected using the forward regression orthogonal decomposition, the smoothing parameter was optimized by the genetic algorithms. Since GAs were used to optimize just the smoothing parameter, the large search space problem mentioned above was alleviated. Of course, our solution might not be optimal due to the suboptimality of the forward regression orthogonal decomposition that was employed in our study, but the algorithm is computationally tractable and often results in a small network structure.

IV. Experiments

A. Experiment 1

In the first experiment, the proposed algorithm was tested using the uniformly distributed data set which was used for the work reported in [3]. There are two classes of data and a total of 2000 data samples were generated for each class, where 500 samples of each class were used for training, the remaining

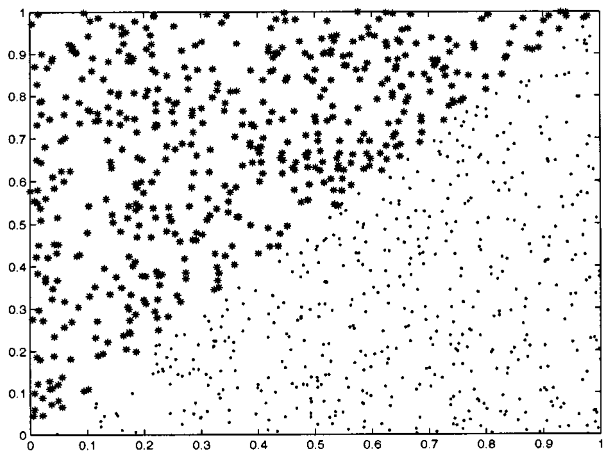
Fig. 3. Training samples for Experiment 1 (·—class 1, ∗—class 2).

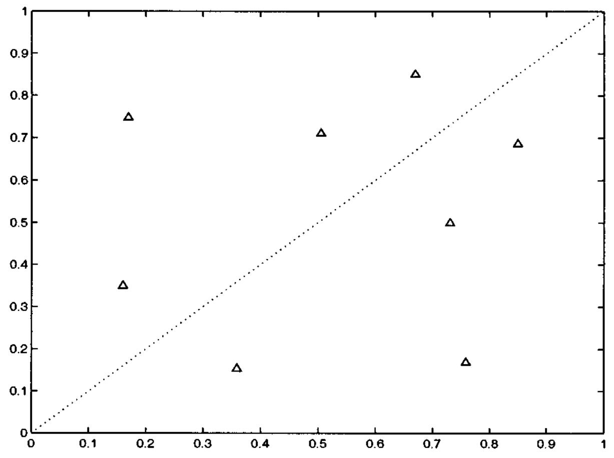
Fig. 4. Selected pattern layer neurons for the Experiment 1.

TABLE I
SELECTED NETWORK STRUCTURE USING THE PROPOSED ALGORITHM FOR EXPERIMENT 1

number of selected neurons	8
smoothing parameter	0.094
correct classification rate (training set)	99.8%
correct classification rate (test set)	98.85%

TABLE II
BEST RESULTS OF LVQ-PNN REPORTED IN [3] FOR EXPERIMENT 1

number of neurons	correct classification rate
10 neurons	96.999%
100 neurons	98.116%

1500 samples were used for test. The feature space is two-dimensional, and the training samples are shown in Fig. 3. The proposed PNN network structure determination algorithm was used to select neurons and the smoothing parameter. The eight selected neurons are shown in [4], the network structure is listed in Table I. Although only eight neurons are used, the PNN classifier achieves 99.8% correct classification for the training data set and 98.85% correct classification for the test data set. For a comparison, the best results of the (LVQ)-PNN reported in [3] are listed in Table II, where 96.999% and 98.166% correct classification were achieved for classifiers with ten and 100 pattern layer neurons, respectively. It was not mentioned in [3] whether the classification process was carried out based on the training data set or the test data set or a mixture of both. But even with only eight neurons and using the test, but not training, data set, our algorithm achieves a relatively better classification result, as well as offering an automatic selection of the smoothing parameter (for the LVQ-PNN method [3], the smoothing parameter has to be selected by trial and error).

An interesting phenomenen in this example is that the selected pattern layer neurons are approximately symmetrical about the boundary of the two classes as shown in Fig. 4. Indeed, if the selected neurons are exactly symmetrical, 100% correct classification can be achieved over both training and test data sets. This result shows that the PNN based classifier is quite effective to solve linear boundary classification problems (it was reported in [3] that PNN was not effective to classify classes with linear boundary).

TABLE III
SELECTED NEURONS USING THE PROPOSED ALGORITHM FOR EXPERIMENT 2

No.	feature 1	feature 2	feature 3	feature 4	class
1	5.0	3.3	1.4	0.2	1
2	6.1	3.0	4.6	1.4	2
3	5.8	2.7	5.1	1.9	3

B. Experiment 2

In the second experiment, the popular Iris data set at UCI Machine Learning Repository (http://www.ics.uci.edu/\~mlearn/MLRepository.html) was used to test the proposed algorithm. The Iris data set consists of 150 samples of three classes, where each class has 50 samples. The dimension of the feature space is four. The 150 samples were equally divided into two subsets. One subset was used for training and the another subset was used for test.

By using our algorithm, only three neurons listed in Table III were selected, where each class had one neuron. The PNN classifier constructed using the three neurons achieved a result of only one misclassification among the 150 samples, where the number of misclassification over the training data and the test data are one and zero, respectively. The automatically selected smoothing parameters for the three classes were 0.2221, 0.2503, and 0.2791, respectively.

Comparisions between our algorithm and other commonly used algorithms such as the LVQ, the fuzzy LVQ called GLVQ-F, the heuristic LVQ called dog-rabbit (DR), the random search (RS) and the GAs were performed. The best results achieved by using our algorithm and those obtained by using other algorithms reported in [8] are listed in Table IV. Notice that in [8] the 150 samples were used for both training and test.

From Table IV, we can see that the PNN classifier having only three pattern layer neurons achieved the result of only one misclassification. This result is better than those of nearest prototype classifiers using the same number of prototypes selected by LVQ, GLVQ-F and DR, whose number of misclassification are 17, 16, and ten, respectively. Our result is also slightly better

TABLE IV
NUMBER OF MISCLASSIFICATION OF DIFFERENT ALGORITHMS WITH THE SAME NUMBER OF SELECTED PROTOTYPES FOR EXPERIMENT 2

algorithm	number of selected samples	number of misclassification
LVQ	3	17
GLVQ-F	3	16
DR	3	10
RS	3	2
GAs	3	2
our algorithm	3	1

TABLE V
NUMBER OF PROTOTYPES NEEDED TO ACHIEVE ONLY ONE MISCLASSIFICATION FOR EXPERIMENT 2

algorithm	number of samples selected	number of misclassification
LVQ	>9	1
GLVQ-F	>9	1
DR	>9	1
RS	7	1
GAs	8	1
our algorithm	3	1

than those achieved by using RS and GAs, whose number of misclassification are both two.

On the other hand, to achieve the result of only one misclassification, our algorithm selected only three neurons, while GAs and RS based algorithms selected eight and seven prototypes, respectively, as shown in Table V. It was not reported in [8] the exact number of prototypes that would be selected to achieve the result of only one misclassification for LVQ, GLVQ-F, and DR based algorithms, but the number would be larger than nine.

The above results are not surprising, since our algorithm configures the PNN classifier by minimizing the number of pattern layer neurons under the constraint of meeting required classification accuracy, while LVQ, GLVQ-F and DR employ other optimization criterion. Beacuse the PNN has additional adjustable parameters (smoothing parameters), our algorithm selects less prototypes than RS and GA-based algorithms.

C. Experiment 3

Face recognition is a typical pattern classification problem. Face image database of the University of Bern (ftp://iamftp.unibe.ch/pub/Images/FaceImages/) was used in this experiment. The database contains frontal views of 30 people, where each person has ten face images corresponding to different head positions. The image size is of 512 $\times$ 342 pixels.

A typical face recognition system consists of three modules, namely, face detection, feature extraction and classification. Since the main concern in the present study is the classification algorithm, it was assumed that face images of 64 $\times$ 48 pixels, as illustrated in Fig. 5, had been detected manually or by automatic face detection algorithms. The face images of 64 $\times$ 48 pixels were downsized to 16 $\times$ 12 pixels by uniform subsampling. The downsized face images as illustrated in Fig. 6 were used as features for classification in the present study.

The face images were represented by matrices. To apply the proposed algorithm, the pattern features are required to put in a vector. Assume that x_{ij} denotes the greylevel of the pixel at ith

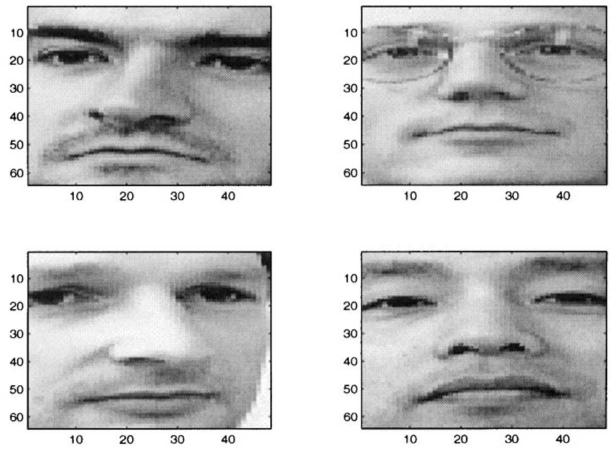
Fig. 5. Face image examples for Experiment 3.

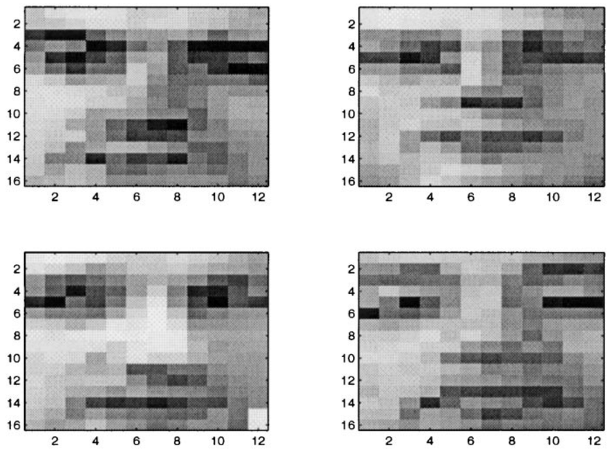
Fig. 6. Downsized face image examples for Experiment 3.

row, jth column of the downsized face image, the feature vector was formed as follows:

$$V = [x_{1,1}, x_{1,2}, \cdots, x_{1,12}, x_{2,1}, x_{2,2}, \cdots \\ x_{2,12}, \cdots, x_{16,1}, x_{16,2}, \cdots, x_{16,12}]^T.$$

Obviously, the face recognition problem under study is a complex pattern classification problem which involves 192 pattern features and 30 pattern classes.

The ten samples in each class were first equally divided into two subsets. Then ten samples were randomly selected from one of the two subsets, and were put into the another subset. The subset that contains seven samples was used for training, the subset that contains three samples was used for test. Totally 210 samples were used for training, 90 samples were used for test.

It was assumed that the number of neurons for each class (person) was the same. The number of neurons and the corresponding classification error rate are listed in Table VI. The diagram of classification error rate (percentage) versus the number of neurons from each class is illustrated in Fig. 7. The smoothing parameter for ecah class is 0.15.

Fig. 7 shows that the classifier achieved zero classification error over the training data and 8.89% error rate over the test data when five neurons were selected for each class. With the increase of the number of neurons, the error rate over the test data

TABLE VI
CLASSIFICATION ERROR RATE AND THE NUMBER OF NEURONS FOR EXPERIMENT 3

number of neurons for each class	error rate (training data)	error rate (test data)
2	26.67%	30%
3	10.95%	15.56%
4	2.86%	11.11%
5	0%	8.89%
6	0%	10%
7	0%	10%

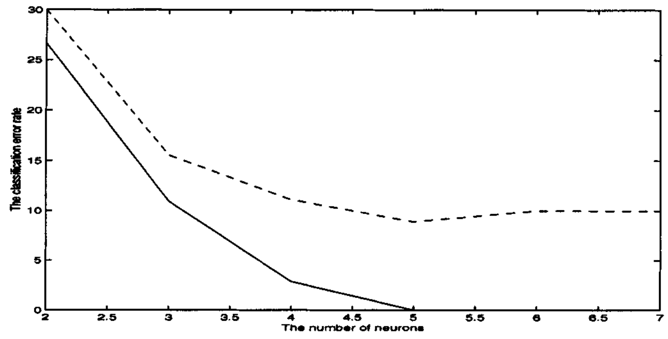

Fig. 7. Classification error rate versus the number of neurons for Experiment 3 (solid—training data, dashed—test data).

is increasing though the error rate over training data is perfect. If the number of neuron is less than five, with the decreasing of the number of neurons, the error rates over both training data and test data are increasing. Therefore five is the appropriate number of pattern layer neurons.

As a matter of fact, the ten face images of each person correspond to five head positions, where each position has two images (but the two images are not exactly the same). The probability density function underlying in the ten samples can therefore be approximated using just five samples. Employing less or more than five samples has the potential of underfitting or overfitting. Our experiment results are in coincidence with the above analysis. More importantly, our algorithm has found the five most representative samples.

V. CONCLUDING REMARKS

Despite considerable progress in probabilistic neural networks, there has been a room for improvement as far as network structure determination is concerned. In this study, a supervised PNN structure determination algorithm has been proposed. A crucial feature of this supervised learning algorithm is that the requirements on the network size and classification error rate are directly incorporated in the process of determining network structure. As a consequence, the proposed algorithm often leads to a fairly small network structure with satisfactory classification accuracy.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for the invaluable comments and the University of Califonia, Irvine, and the University of Bern, Switzerland, for maintaining the Iris data set and the face image database that were used in Experiments 2 and 3, respectively.

REFERENCES

[1] S. A. Billings, S. Chen, and M. J. Korenberg, “Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator,” *Int. J. Contr.*, vol. 49, pp. 2157–2189, 1988.
[2] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press, 1995.
[3] P. Burrascano, “Learning vector quantization for the probabilistic neural network,” *IEEE Trans. Neural Networks*, vol. 2, pp. 458–461, July 1991.
[4] 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, pp. 302–309, Mar. 1991.
[5] S. Chen, Y. Wu, and B. L. Luk, “Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks,” *IEEE Trans. Neural Networks*, vol. 10, pp. 1239–1243, Sept. 1999.
[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[7] C. Kramer, B. Mckay, and J. Belina, “Probabilistic neural network array architecture for ECG classification,” in *Proc. Annu. Int. Conf. IEEE Eng. Medicine Biol.*, vol. 17, 1995, pp. 807–808.
[8] L. I. Kuncheva and J. C. Bezdek, “Nearest prototype classification: Clustering, genetic algorithms, or random search,” *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 160–164, Feb. 1998.
[9] S. Ma and C. Ji, “Performance and efficiency: Recent advances in supervised learning,” *Proc. IEEE*, vol. 87, pp. 1519–1535, 1999.
[10] M. T. Musavi, K. H. Chan, D. M. Hummels, and K. Kalantri, “On the generalization ability of neural-network classifier,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 6, pp. 659–663, 1994.
[11] R. D. Romero, D. S. Touretzky, and G. H. Thibadeau, “Optical Chinese character recognition using probabilistic neural networks,” *Pattern Recognit.*, vol. 3, no. 8, pp. 1279–1292, 1997.
[12] P. P. Raghu and B. Yegnanarayana, “Supervised texture classification using a probabilistic neural network and constraint satisfaction model,” *IEEE Trans. Neural Networks*, vol. 9, pp. 516–522, May 1998.
[13] D. F. Specht, “Probabilistic neural networks,” *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.
[14] ——, “Enhancements to the probabilistic neural networks,” in *Proc. IEEE Int. Joint Conf. Neural Networks*, Baltimore, MD, 1992, pp. 761–768.
[15] R. L. Streit and T. E. Luginbuhl, “Maximum likelihood training of probabilistic neural network,” *IEEE Trans. Neural Networks*, vol. 5, pp. 764–783, Sept. 1994.
[16] Y. N. Sun, M. H. Horng, X. Z. Lin, and J. Y. Wang, “Ultrasonic image analysis for liver diagnosis-a-noninvasive alternative to determine liver disease,” *IEEE Eng. Med. Biol. Mag.*, vol. 15, no. 1, pp. 93–101, 1996.
[17] H. G. C. Traven, “A neural-network approach to statistical pattern classification by semiparametric estimation of a probability density functions,” *IEEE Trans. Neural Networks*, vol. 2, pp. 366–377, May 1991.
[18] L. X. Wang and J. M. Mendel, “Fuzzy basis functions universal approximation, and orthogonal least squares learning,” *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, Sept. 1992.
[19] L. Wang and R. Langari, “Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques,” *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 454–458, 1995.
[20] B. A. Whitehead and T. D. Choate, “Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction,” *IEEE Trans. Neural Networks*, vol. 7, pp. 869–880, July 1996.
[21] A. Zaknich, “A vector quantization reduction method for the probabilistic neural network,” in *Proc. IEEE Int. Conf. Neural Networks*, Piscataway, NJ, 1997, pp. 1117–1120.