The Structure Optimization of Radial Basis Probabilistic Neural Networks Based on Genetic Algorithms

Wenbo Zhao[1,2] De-Shuang Huang[2] Ge Yunjian[2]

1. Department of Automation, University of Science and Technology of China
2. Institute of Intelligent Machines, Chinese Academy of Sciences.
P.O.Box 1130, Hefei, Anhui , China.

(This work was supported by Grants 60173050 from NSF of China)

Abstract- In this paper, the genetic algorithm is introduced into the structure optimization of radial basis probabilistic neural networks. A special encoding method of individuals is proposed in this paper. It can be found that the encoding method involves not only the number but also the locations of selected hidden centers. In addition, for this encoding method we construct a fitness function for precision control. To speed up the training we employ the method of matrix pseudoinverse to train the corresponding networks. Finally, we take the telling-two-spirals-apart problem for example to validate the ability of the genetic algorithm for the structure optimization of radial basis probabilistic neural networks.

I. INTRODUCTION

The radial basis probabilistic neural network (RBPNN) model proposed by Huang [1], shown in Figure.1, comes from the radial basis function neural network (RBFNN) and the probabilistic neural network (PNN). So it possesses the advantages of the above two networks. But it lowers the demerits of the two networks. Compared with RBFNN, the requirement for the training and the testing of RBPNN is obviously reduced. Moreover, it has a smaller structured network compared with PNN. As shown in Figure.1, this network consists of four layers: one input layer, two hidden layers and one output layer. The first hidden layer is a nonlinear processing layer, generally consisting of selected hidden centers determined by input training set. The second hidden layer corresponds with the first hidden layer, which has generally the same size as the output layer for a labeled pattern classification problem. The second hidden layer makes the selection on the first hidden neurons and sums the selected first hidden outputs for its own output. In general, the corresponding second hidden layer weights for RBPNNs are 1s. The last layer is just the output layer.

In mathematic terminology, the ith output of RBPNN can be written as

$$y_i = \sum_{k=1}^{m} w_{ik} H_k(x) \quad i = 1,2,\cdots m \tag{1}$$

$$H_k(x) = \sum_{i=1}^{n_k} \phi_i(x, c_{ki}) = \sum_{i=1}^{n_k} \phi_i(\|x - c_{ki}\|_2) \quad k = 1,2,\cdots m \tag{2}$$

Where x is a n-dimensional input vector, $H_k(x)$ is the second hidden layer output; w_{ik} is the weight from the kth hidden neuron of the second hidden layer to the ith output neuron; $\phi_k(\cdot)$ is the nonlinear mapping function (or kernel function) of the first hidden layer; c_{ki} is the ith hidden center vector corresponding to the kth hidden neuron of the second hidden layer and $\|\cdot\|_2$ denotes the Euclidean norm.

Assuming that the total number of the training samples is N, based on (1) we can write the vector-matrix form of some output neuron as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & h_{2m} \\ \vdots & \vdots & & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Nm} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \tag{3}$$

Or

$$Y = HW \tag{4}$$

Like RBFNN, the selection of hidden centers of the RBPNN is very important to improve the performance of networks. If all the training samples are selected as the hidden centers, although the network being guaranteed to converge to some satisfying solution, the generalization capability of the network will become very poor so that many noised or deformed samples will be not able to be recognized. Therefore, in this paper, our main task is to discuss how to optimize the structure of the RBPNN, i.e., to determine the hidden neuron number and the hidden center locations of the first hidden layer of the RBPNN by genetic algorithm.

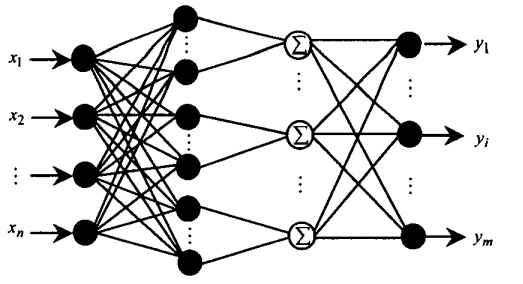


Figure.1 The structure of radial basis probabilistic neural networks

The genetic algorithm is a kind of global search method, which takes full advantage of the selection operator, the

crossover operator and the mutation operator to realize the optimization selection and the computation by means of simulating evolution of natural world. It depends on several key points such as encoding, constructing the fitness function, genetic operators, selecting algorithms and so on. Conveniently we introduce some symbols to express our statements as follows.

Assume $S=\{0,1\}^l$, S^N, S^2, $\vec{X}=(X_1,X_2,\cdots,X_N)$ respectively represents an individual space, a population space, a parent space and a population, where l, N, X_i respectively denotes the length, the population size and the ith individual of the population. We use $\vec{X}(0),\cdots$, and $\vec{X}(t)$ express the initial population, $\cdots$, and the nth generation population. So the population sequence $\vec{X}(0),\cdots$, and $\vec{X}(t)$ is a Markov chain according to [2].

II. Main Results and Experiments

Generally speaking, it is very important for genetic algorithm to encode an individual, construct the fitness function and select the genetic operators. This paper is focused on applying the genetic algorithm to selecting hidden centers or optimizing the structure of the RBPNN. First, we present several related concepts of the genetic algorithm.

A. Encoding of an Individual

From the above analyses, we know the key point of designing the RBPNN is to select the hidden centers of the first hidden layer. Generally, the selection of the hidden centers in the first hidden layer of the RBPNN is not only involved in the number of the hidden centers, but also in the space locations of the hidden centers. Usually, we wish the number of the hidden centers to be small enough for the fewer hidden centers will not only simplify the training and the testing of the network, but also improve the generalization capability of the network. On the other hand, the locations of the hidden centers in space are of utmost importance to the performance of the network. In the case of the fixed number of the hidden centers, different locations for the hidden centers can lead to different performance of the network. Based on these considerations, we propose a new encoding method.

As shown in Figure 2, we assume all the training samples to be chosen as the genes of an individual in the case of smaller number of training samples. Actually when the number of training samples is very large, the situation that all the training samples are selected as initial hidden centers will require large computation consumptions. So some heuristic methods, e.g., *k-means clustering,* can be used to help initially select hidden centers. From Figure 2, we can see that for each gene, which only represents a fixed sample, we make use of the binary symbols (0 or 1) to denote the existence of a training sample (or initially selected) in the chosen hidden centers. That is, the binary signal "0" represents a sample not to be selected while "1" represents the existence of a training sample point. Therefore, each individual is made of the sequential symbols of "0" and "1". In other words, the genes in an individual can correspond with the training samples one by one. As a result, the summation of "1's" in each individual represents the length of the selected hidden centers. From the viewpoint of the population each individual has its own length. So the encoding method integrates considering the information of the number and the locations in space of hidden centers.

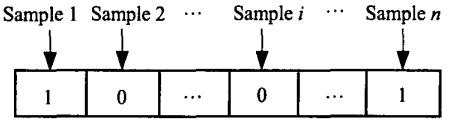


Figure.2 Encoding of individual of hidden centers

B. Construction of the fitness function

In general, the fitness function plays an important role in genetic algorithm, which can be expressed as a mapping $f:S\to R^+$ in mathematics, where R^+ denotes the positive real number space. Our intention is to make the number of hidden centers as few as possible selected under the given precision. So while constructing the fitness function we must consider, at the same time, the number of the hidden centers and the precision and the error information. According to the theorem of the maximum fitness function, the fitness must be monotonously decreased with the increase of the number of the hidden centers, where the practical errors will excess the given errors and the fitness must reach the minimum (mostly zero) regardless of the number of the hidden centers. Based on these considerations we define the fitness function as follows.

$$f(\varepsilon,n_c)=\frac{(b+1)N}{2^b n_c} \tag{5}$$

$$b=sign(\varepsilon-e) \tag{6}$$

where ε represents the given error criterion; e is the practical error; N is the total number of training samples; and n_c represents the number of selected hidden centers.

According to the statement in section A, n_c can be described as follows.

$$n_c=\sum_k^l X_i(k) \quad n_c\in(1,N) \tag{7}$$

where X_i denotes the *ith* individual of population; l represents the individual length.

In the definition of the fitness the actual errors e is an unknown parameter. Now we introduce a special method to solve this parameter. Before introduction, we must introduce

the concept of the error cost function defined in the outer-supervised learning feedforward neural networks (FNN's). For most outer-supervised learning FNN's, the usually used optimization criterion is the *mean-squares-error* between the actual and the desired network output. Therefore, according to this criterion here for RBPNN the error cost function is defined as

$$J(W)=\frac{1}{2}\|e\|_2^2=\frac{1}{2}\sum_{k=1}^{m}[y_d(k)-y(k)]^2 \qquad (8)$$

where m denotes the number of the second hidden neurons or the output neurons.

Further, (8) can be rewritten as a vector–matrix form

$$J(W)=\frac{1}{2}(Y_d-Y)^T(Y_d-Y) \qquad (9)$$

where Y_d denotes the matrix consisting of the desired outputs, and Y the matrix composing of the actual outputs of the network.

Substituting (4) into (9), there gives

$$\begin{aligned} J(W)&=\frac{1}{2}(Y_d-HW)^T(Y_d-HW)\\ &=\frac{1}{2}(Y_d^TY_d-2Y_d^THW+W^TH^THW) \end{aligned} \qquad (10)$$

Minimizing the cost function $J(W)$, we obtain

$$\frac{\partial J(W)}{\partial W}=0 \qquad (11)$$

That is

$$-H^TY_d+H^THW=0 \qquad (12)$$

So we can derive the solution of W

$$W=(H^TH)^{-1}H^TY_d=H^+Y_d \qquad (13)$$

where H^+ denotes the Moore-Penrose pseudoinverse of the output matrix of the second hidden layer.

Since the weights of the output layer have been obtained, the actual output becomes into

$$Y=HW=HH^+Y_d \qquad (14)$$

So we can obtain the error e from (10) and (14)

$$e=\left\|Y_d-HH^+Y_d\right\|_2^2 \qquad (15)$$

For classification problem, (15) can be expressed as follows.

$$e=\left\|Y_d-round(HH^+Y_d)\right\|_2^2 \qquad (15b)$$

where $round(\bullet)$ refers to rounding the elements of bracket $(\bullet)$ to the nearest integers. At this time if desired signals are set as a matrix consisting of unitary vectors, e might be $0,1,4,8,16,\cdots$ and so on.

According to the theorem of M-P pseudoinverse, we know that W is a least-square and least norm solution satisfying (12). Generally, there exist many solutions satisfying (12), but only the minimized norm solution W is chosen, which assure (8) or (9) to approach the minimum. Actually, if the hidden centers are not suitably chosen, (8) or (9) will be greater than zeros. From (15) it can be seen that in the procedure of computing fitness of individuals we do not compute the weights of the output layer apart from finding H^+.

In the above procedure, it can also be seen that while solving e the key point is to compute the pseudoinverse H^+. There are some software tools such as MATLAB, etc., which might be used. On the other hand, we can do it well by means of the method introduced in [8].

C. Selection of algorithms and Construction of Genetic Operators

The selection of genetic algorithms can make an effect on the convergence performance for the real problems since the poor selected algorithm can lead to very long training time to approach the optimal solution. Sometimes the expected optimal solution could not be obtained in bad initial population. According to [2], there exist three kinds of genetic algorithms to be used, which are the canonical genetic algorithm, the elitist selection genetic algorithm, and the steady state genetic algorithm. From [2], we know that Markov sequences $\{\bar{X}(0),\bar{X}(1),\cdots,\bar{X}(t)\}$ produced by the elitist selection genetic algorithm will converge to a subset M_0^* of the optimal population set M^* by probability 1, which can be in mathematics expressed as follows

$$\lim_{t\to\infty}P\{\bar{X}(t)\in M_0^*/\bar{X}(0)\}=1 \qquad (16)$$

Therefore, the elitist selection genetic algorithm is adopted in the structure optimization of RBPNN.

Generally, the selection and construction of genetic operators are very important for the genetic algorithm to converge within a limited time and iterations. For the genetic algorithms, the standard genetic operators consist of three parts: the selection operator, the crossover operator and the mutation operator. Here we in detail introduce these operators as follows.

1) Selection operator: It consists of two aspects in our algorithm. The one is to select the optimal individual in the current population based on the fitness of each individual of population. In other words we select the individual with the maximum fitness of the current population as a component of next generation without crossover and mutation operation. The other aspect is to select the mating parent of next generation.

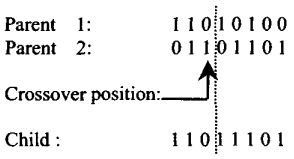
In our algorithm the selection operator of mating parent is based on the selection probability of each pair parent

individuals in current generation. The selection probability can be calculated as follows

$$P\{X_i\} = f(X_i) / \sum_{i=1}^{N} f(X_i) \tag{17}$$

The usual selection operator is the roulette wheel selection. Its advantage is that the individual with low fitness has a chance to be selected into a new generation whereas those good individuals with high fitness have more chances than those low ones.

2) Crossover operator: Crossover operator elevates in essence the genetic algorithm. There are several crossover operators such as single-point crossover operator, single-point-stochastic crossover operator, double-points or multi-points crossover operator and uniform-stochastic crossover operator, and so on. Considering possible large size of training samples, in our algorithm the single-point stochastic crossover is adopted because of its smaller number of computations and easy realization. The point, where the crossover operator is executed, is usually the stochastic selection between the second gene and the most neighbouring gene to the end gene of an individual. According to crossover probability P_c, we substitute the genes between crossover point and the end of parent 2 into the corresponding part of parent 1. As a result, a temporary child individual is produced. The principle is shown in Figure 3.



3) Mutation Operator: The mutation operator is always used to keep the diversity of population. It independently employ the logical *not* operation by mutation probability P_m. Generally the mutation probability is set a very small value, usually chosen in the range of [0.001, 0.01]. In essence the mutation operator can improve the local search capability, which can guarantee the genetic algorithm to converge to the optimal solution in its neighbouring field. Generally each gene of the above temporary child individual produced by the crossover operator is employed to produce the mutation operation by mutation probability. Hence, a child individual of new generation is produced.

4) Procedure of GA: According to the above statements, we can obtain the GA procedure. The details of the procedure are summarized in Table I.

TABLE I

THE STEPS OF GA FOR SELECTING THE HIDDEN CENTERS OF RBPNN

Step 1.Stochastically initialize the population $\vec{X}(t), t = 0$, t denotes the index of generation.

Step 2.Calculate the fitness (f_i i = 1,2,⋯N) of individuals in population $\vec{X}(t)$, N is the size of population. According to the fitness calculate the selection probability P_{si}. $i = 1,2,\cdots,N$

Step 3.Based on the selection probability P_{si}, independently select *N-1* pairs of parent $\left(M_1^k, M_2^k\right)$ $(k = 1 \sim N-1)$ among the population $\vec{X}(t)$ by roulette wheel selection. And search for the maximal fitness individual marked as $X_{i_0}(t) = \arg \max_j \left\{ f(X_j(t)) \right\}$ $j = 1,2,\cdots,N$.

Step 4.For *N*-1 pairs of parent $\left(M_1^k, M_2^k\right)$ $(k = 1,2,\cdots,N-1)$ employ the crossover operator by single-point-stochastic crossover based on the crossover-probability P_c and obtain *N*-1 temporary individuals $X_1'(t+1),\cdots,X_{N-1}'(t+1)$, and for these temporary individuals employ mutation by mutation-probability P_m, then obtain *N*-1 new individuals generation $X_1(t+1),\cdots,X_{N-1}(t+1)$

Step 5. According to Step2 and Step 4 produce the next population. $\vec{X}(t+1) = \{X_1(t+1),\cdots,X_{N-1}(t+1), X_N(t+1)\}$ where $X_N(t+1) = X_{i_0}(t)$.

Step 6. If the stop condition is satisfied, stop computing; else go to step 2.

D. *Simulation Results*

In order to test our GA for how to select the hidden centers of the first hidden layer of the RBPNN, we take the telling-two-spirals-apart problem for example to verify our algorithm. The telling-two-spirals-apart is a neural network benchmark test problem proposed by Wieland (Lang and Witbrock, 1989) reported by [5]. The benchmark test is a quite difficult task. In order to classify them, Witbrock and Land (1989) crafted a neural network with special 2-5-5-1 architecture. They trained the network with vanilla backpropagation and QuickProp. As a result, it was found that the network converged to a fixed accuracy after 20,000 and 8000 iterations, respectively.

In our testing, we select 100 points in each spiral. Thus, there are in total 200 training points. The training points $(x^k, y^k, b^k)(k = 1,2,\cdots,200)$ are produced according to the following equations.

$$x^{2n-1} = r_n \sin\alpha_n + 0.5 = 1 - x^{2n} \tag{18}$$

$$x^{2n} = 0.5 - r_n \sin\alpha_n = 1 - x^{2n-1} \tag{19}$$

$$y^{2n-1} = r_n \cos\alpha_n + 0.5 = 1 - y^{2n} \tag{20}$$

$$y^{2n} = 0.5 - r_n \cos\alpha_n = 1 - y^{2n-1} \tag{21}$$

$$b^{2n-1} = 1 \tag{22}$$

$$b^{2n} = 0 \tag{23}$$

$$r_n = 0.4\left(\frac{105 - n}{104}\right) \tag{24}$$

$$\alpha_n = \frac{\pi(n-1)}{25} \tag{25}$$

where (x^k, y^k) represents the points on a plane, and b^k the class of the spiral.

In our experiments we make use of the RBPNN to tell two spirals apart. Input vectors are two-dimensional points (x^k, y^k). There are two output neurons to denote two kinds of spirals. It means that the teacher signal $(1,0)$ is one spiral, and the one $(0,1)$ is the other spiral. Errors are determined by (15b). The kernel function for the first hidden neurons is chosen as the Gaussian function. The variance σ of the kernel function is set according to the following heuristic relationship [4]

$$\sigma = \frac{d_{\max}}{\sqrt{K}} \tag{26}$$

Where $d_{\max}$ is the maximum Euclidean distance among all the training samples, and K the number of the samples. Conveniently we set σ as a fixed value 0.04. The two parameters of genetic algorithms, the crossover probability is selected as 0.8, and the mutation probability as 0.005.

In the experiments, there are 200 points taking part in the training. The termination error ε is set as zero. After 300 iterations, the curve of the number of selected hidden centers and the optimal fitness versus the generation is shown as Figure.5 and Figure.6. By means of two figures we can see that after 195 iterations our algorithm starts to converge. So we obtain the optimal hidden centers. The selected centers among 200 training sample points are shown in Figure.4, i.e., the points with the signal "+", where the hidden centers are chosen as about 16% (32 hidden centers) the original training samples. At this time the actual output of the network has still the good classification performance as shown in Figure.7.

We use the selected hidden centers to verify the generalization capability in two aspects. Firstly we reduce the step size from 1 to 0.1 in (24) and (25). Consequently, there are 1982 points taking part in the testing. The results with 100% separated testing samples are shown in Figure.8. The other method is to test the trained network with noise signals based on the first method. The noises are Gaussian white noises with zeros-mean and three different variances $(0.0001, 0.001, 0.01)$. The three input data with noises are plotted on Figure 9. The test results are shown in Table II. From Figure.9 and Table II, it can be seen that the optimized RBPNN has indeed good performance in generalization capability.

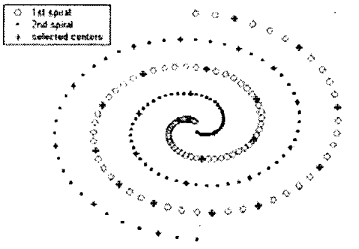


Figure.4 The distribution of selected hidden centers

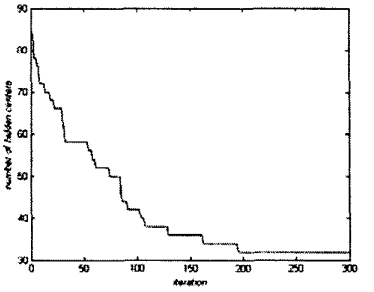


Figure.5 The changing curve of the number of hidden centers *vs* iteration

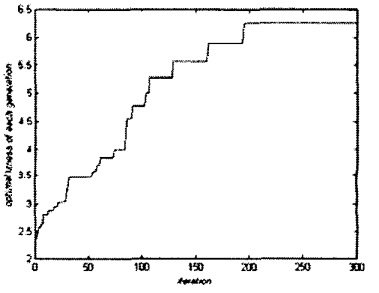


Figure.6 Changing curve of the optimal fitness

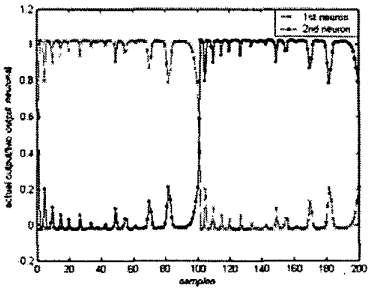


Figure.7 Actual output curve of the output layer

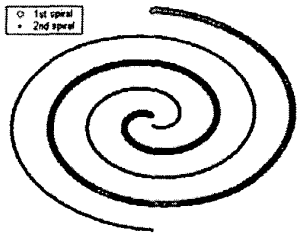


Figure.8 The generalization capability of the networks with 32 centers

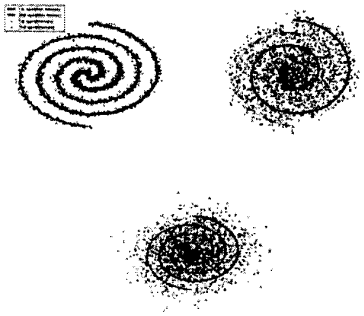


Figure.9 Noises spirals *vs* no noise spirals

TABLE II

THE TESTING RESULTS WITH NOISE SAMPLES

Noise variances	$\sigma = 0$	$\sigma = 0.0001$	$\sigma = 0.001$	$\sigma = 0.01$
Recognized rates (1st spiral)	1	0.9879	0.8123	0.5449
Recognized rates (2nd spiral)	1	0.9879	0.8315	0.5388

E. Conclusions

Genetic algorithm (GA) is a global search method, which has the ability to globally approach the optimization solutions of given problems. In this paper, we propose using the GA to select the hidden centers (including the number and the locations of the hidden centers in space) of the first hidden layer of the RBPNN so as to obtain the optimization performance such as classification reliability and generality capability. Our focus is to design an encoding method and choose an appropriate fitness for optimizing the structure of the RBPNN. The experimental results by telling-two-spirals-apart problem show that the genetic design method for choosing the hidden centers of the first hidden layer of the RBPNN is efficient and effective. Further research works will include discussing more complex real problems and comparing the performance of this GA method with other methods.

F. References

[1] D.S. Huang, "Radial basis probabilistic neural networks: Model and application", International Journal of Pattern Recognition and Artificial Intelligence, 13(7), 1083-1101, 1999.

[2] Zhang Wenxiu, Leung Yee, "Mathematical Foundation of Genetic Algorithms", Xi'an Jiaotong University Press, China, May, 2000.

[3] Ferdric M. Ham, Ivica Kostanic, "Principles of Neurocomputing for Science and Engineering", McGraw-Hill Higher Education, New York, pp.140-152, 2001.

[4] S. Haykin, "Adapitve Filter Theory", 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.

[5] Zhihua Zhou, Shifu Chen, Zhaoqian Chen, "FANNC: A Fast Adaptive Neural Network Classifier", Knowledge and Information Systems, pp 115-129, Feb. 2000.

[6] Zekeriya Uykan, Cüneyt Güzelis, M. Ertugrul Celebi, Heikki N. Koivo, "Analysis of Input-Output Clustering for Determing Centers of RBFN", IEEE Transactions on Neural Networks, Vol.11, NO. 4, July 2000.

[7] Guo-Liang Chen, Xu-Fa Wang, Zhen-Quan Zhuang, Dong-Sheng Wang, "Genetic Algorithms and Application", Publishing House of People Post-Communication, Beijing, China, 1996.

[8] Guo-Rong Wang, "Matrix and Operators of Generalized Pseudoinverse", Publishing House of Science, Beijing, China, pp. 95-130, June, 1994.

[9] Zheng-Jun Pan, Li-Shan Tang, Yu-Ping Chen, "Evolutionary Computation", Publishing House of University of Tsing Hua, Publishing House of Science & Technology of Guang Xi, China, pp.9-37, July, 1998.