Parallel Probabilistic Neural Network (PPNN)

Bo Xu[1]
Indiana University

Liqing Zheng
Purdue University

Abstract----It was pointed out in this paper that the planar topology of current back-propagation neural network (BPNN) sets limits to solve the slow convergence rate problem, local minima, incapability of learning etc. problems associated with BPNN. The parallel probabilistic neural network (PPNN) using a new neural network topology----stereotopology----was proposed to overcome these problems. The learning ability and the generalization ability of BPNN and PPNN were compared for several problems. The simulation results show that PPNN was capable of learning any kinds of problems much faster than BPNN and generalized better than BPNN too. It was analyzed that the faster, universal learnability of PPNN was due to the *parallel* characteristic of PPNN's stereotopology, and the better generalization ability came from the *probabilistic* characteristic of PPNN's memory retrieval rule PMRR.

Key Words ---- Back-propagation neural network (BPNN), parallel probabilistic neural network (PPNN), planar topology, stereotopology, probabilistic memory retrieval rule (PMRR), learning ability, generalization ability, representations.

1. INTRODUCTION

In the present neural network area, the back-propagation neural network (BPNN) (Werbos, 1974; Parker, 1985; le Cun, 1985; Rumelhart, Hinton, and Williams, 1986) is perhaps the most widely used neural net. However, BPNN has its drawbacks: (1). Slow rate of learning. BPNN has a very slow rate of convergence for most applications. It usually takes hundreds of thousands of times (epochs) in training for BPNN to converge at a specific accuracy. (2). Local minima. There are many local minima in the cost function curve (MSE vs weights) of BPNN for most applications. These local minima may reduce BPNN's convergence rate or even prevent it from reaching the global minimum. (3). Incapability of learning. For some very difficult applications, the global minimum of BPNN is so high that even if the local minima problem is overcome and the global minimum is reached, the preset requirement for learning accuracy still cannot be met. So these problems in fact cannot be learned by BPNN at the accuracy we designed beforehand. These drawbacks greatly hamper the practical application of BPNN.

Many attempts have been made to increase BPNN's rate of convergence (Austin, 1988; Fahlman, 1988; Hayduk and Fong, 1988; Pao and Beer, 1988; Jacobs, 1988;

[1]We thank Drs. R. Rhoades, L. Knoebel, D. Peavy, P. G. Madhavan, D. Suzuki, H. Oner Yurtseven, and A. Sinha for their supports, thank Dr. S. Ochs for enlightenment discussion and encouragements, thank Dr. L. Abels for providing reference materials. We are grateful to Drs. S.E. Fahlman and Y. Chen for valuable discussions and comments on the manuscript.

Shepanski, 1988; Fogaca, Kramer, and Moore, 1988; Winter and Widrow, 1988; Vogl, Mangis, Rigler, *et al*., 1988; Peterson and Hartman, 1989; Tollenaere, 1990; Shawe-Taylor and Cohen, 1990; Brent, 1991; Madhavan, Xu, Stephens, and Low, 1991; Rigler, Irvine, and Vogl, 1991) and to overcome the local minima problem (Ackley, Hinton, and Sejnowski, 1985; Cuykendall and Reese, 1988; Baba, 1989; Baldi and Hornik, 1989; Hirose, Yamashita, and Hijiya, 1991). These studies have shown encouraging improvements to the original BPNN.

The architecture of neural network consists of three parts (Lippmann, 1987): (1) learning algorithms of weights; (2) characteristics (activation functions) of nodes; and (3) topology of the network. Most of the above attempts fall into the first two parts: the learning algorithms and the characteristics of nodes.

Although there are many improvements in increasing the rate of convergence and overcoming the local minima problems through the study of learning algorithms and/or the node characteristics, we proposed in (Xu and Zheng, 1991) that it is the third part, the topology of BPNN (a planar topology), that sets limits for the improvements of BPNN's performances.

This paper is a detailed version and further development of above IJCNN paper on PPNN. The main purpose of PPNN is trying to overcome the slow rate of convergence, local minima, and the incapability of learning problems associated with BPNN by using a new neural network topology----stereotopology, and a new memory retrieval method----PMRR.

2. PPNN STEREOTOPOLOGY

In order to see the features of PPNN's new stereotopology, let's first look at BPNN's topology (Figure 1). There is only one branch of hidden layer(s) connecting with the input and the output layers in BPNN. From the side view, the input layer, the output layer, and all the hidden layers can lie in one plane. Therefore, we say BPNN has a *planar* topology.

The key feature of PPNN is that there are multi-branches of parallel hidden layers in it. The topology of PPNN is shown in Figure 2. PPNN also has an input layer and an output layer. However, unlike the BPNN which has only one branch of hidden layer, the PPNN has P (the number of training patterns) parallel branches of hidden layers. There is no connection between these parallel branches of hidden layers. Each of the P parallel branches of hidden layers connects directly to the input layer and the output layer in the same way as the hidden layer in BPNN. From the side view, the input layer, output layer, and all the P parallel branches of hidden layers cannot lie in one plane. Therefore, we say PPNN has a *stereotopology*.

3. PPNN TRAINING PROCESS

There is a one-to-one correspondence between the P parallel branches of hidden layers and P training patterns. Each of the P parallel branches of hidden layers corresponds to each of the P training patterns.

Each of the P parallel branches of hidden layers can be in either on or off state. The *on-state* is the state in which this branch of hidden layers is connected with the input layer and the output layer. This is the working state in which this branch of

hidden layers takes part in the training or testing processes of PPNN. The *off-state* is the state in which this branch of hidden layers is disconnected with both the input layer and the output layer. This is the resting state in which this branch of hidden layers does not take part in the training or testing processes of PPNN.

During training, only one of the P parallel branches of hidden layers is in the on-state. All the other branches of hidden layers are in the off-state. Which branch of hidden layers should be on during training is determined by the corresponding training pattern. If the training pattern p (p=1,2,3, ... , P) is used to train the PPNN, then the pth branch of hidden layers should be on, and all the other branches of hidden layers should be off.

When the pth training pattern is used and the pth parallel branch of hidden layers is on, the weights updating procedure for the pth branch of hidden layers is the same as that in BPNN. All the weights are updated with back-propagation algorithm in order to minimize the cost function.

4. PPNN TESTING PROCESS----PMRR

In BPNN, because there is only one branch of hidden layers, it is not a problem to decide which branch of hidden layers should be on during testing and memory/information retrieval process.

However, in PPNN, there are P parallel branches of hidden layers, and each of them can be in either on or off state. So which of the P parallel branches of hidden layers should be on, and how they contribute their memories stored within their weights to the net test output become problems to be resolved first.

Here we propose a *probabilistic memory retrieval rule* (PMRR) for the PPNN testing process and answer these questions.

4.1. Closeness Selection

First, the test pattern is compared with each of the P training patterns to find out the training pattern p' which is closest to the test pattern. To do so, we calculate the total absolute difference between the test inputs and each of the P training patterns' inputs. Denoting

I_i^{test} ---- the input value of the test pattern on the input node i;

I_{pi}^{train} ---- the input value of the pth training pattern on the input node i;

N^{in} ---- the total input nodes number.

Then the total absolute difference Δ_p between the test inputs and the inputs of the pth training pattern is

$$\Delta_p = \sum_{i=1}^{N^{in}} \left| I_i^{test} - I_{pi}^{train} \right| \qquad (1)$$

The summation is over the input layer.

Using p' to denote the training pattern which is closest to the test pattern. Then we have

$$\Delta_{p'} = \min(\Delta_p) \tag{2}$$

4.2. Probabilistic Memory Retrieval Rule (PMRR)

Once the training pattern p' is determined, it is needed to check whether this p' is close enough to the test pattern. To do so, a closeness index ε is preset (ε can be set to any small positive number, depending on the precision demanded by the application). The ε is used as a criterion to check the closeness between the training pattern p' and the test pattern. Denoting

P ---- total training pattern number;
r ---- closeness ratio which is defined as

$$r = \frac{\Delta_{p'}/N^{in}}{\sum_{p=1}^{P}\sum_{i=1}^{N^{in}} \left|I_{pi}^{train}\right|/P*N^{in}} \tag{3}$$

where the summation for i is over the entire input layer and the summation for p is over all the training patterns. Then if

$$r \le \varepsilon \tag{4}$$

is true, we say that the training pattern p' is close enough to the test pattern at the closeness ε. Otherwise, if

$$r > \varepsilon \tag{5}$$

we say that the training pattern p' is not close enough to the test pattern at ε.

Based on the closeness of p' to the test pattern, the memory retrieval processes are as follows:

(**i**). **If training pattern p' is not close enough to the test pattern at** ε.

In this case, from (2) and (5) we conclude that

$$\Delta_p \ge \Delta_{p'} > 0 \tag{6}$$

Denoting

O_{pj} ---- the test output value on the jth output node produced by the pth parallel branch of hidden layers alone (i.e. the test output produced when only the pth parallel branch of hidden layers is on during testing);

O_j ---- the net test output value on the jth output node produced when all P parallel branches of hidden layers are on.

Then we propose that the O_j should be the weighted mean over O_{pj}:

$$O_j = \frac{\sum_{p=1}^{P} a_p O_{pj}}{\sum_{p=1}^{P} a_p} \tag{7}$$

where the a_p is the weighted mean's coefficients.

The value of a_p depends on the total absolute difference Δ_p between the training pattern p and the test pattern. The relationship between a_p and Δ_p should be inversely proportional to each other because the closer the training pattern p is to the test pattern (indicated by smaller Δ_p), the larger the pth parallel branch of hidden layers should contribute to the net output (indicated by larger a_p).

We have studied several functions for the relation between a_p and Δ_p. We used parabolic problem to train and test PPNN with these different a_p functions. The results were compared in Figure 3. Noting that (6) guarantees that the denominators of a_p will not be zero here.

From the comparison in Figure 3, we see that the inverse square power weighted coefficients

$$a_p = \frac{1}{\Delta_p^2} \tag{8}$$

gives the best performance for generalization ability. Following, we will get a_p according to (8) whenever it is needed.

From the probability viewpoint, the

$$f_p = \frac{a_p}{\sum_{p=1}^{P} a_p} \tag{9}$$

is just the probability that the pth parallel branch of hidden layers will contribute to the net test output. This is the reason we call this kind of memory recall method *probabilistic memory retrieval rule* (PMRR).

(**ii**). **If training pattern p' is close enough to the test pattern at** ε.

In this case, there is the possibility that

$$\Delta_p = 0 \tag{10}$$

So (8) will not be valid any more.

However, because ε can be any small positive number, it was so chosen that when (4) is true, it is reasonable to assume that the p'th parallel branch of hidden layers alone will contain enough information for the net test output. Therefore, it is sufficient to use only the p'th parallel branch of hidden layers for test (i.e. only p'th branch is on, all the other branches of parallel hidden layers are off). So the net test output now is simply

$$O_j = O_{p'j} \tag{11}$$

(11) can also be regarded as the limit of (7) as $\Delta_{p'}$ tends to zero or $a_{p'}$ tends to infinity:

$$\lim_{\Delta_{p'}\to 0} O_j = \lim_{a_{p'}\to \infty} \frac{\sum_{p=1}^{P} a_p O_{pj}}{\sum_{p=1}^{P} a_p} = O_{p'j} \tag{12}$$

which makes the p'th parallel branch of hidden layers having a much larger weighted mean coefficient $a_{p'}$ than all the other branches' weighted mean's coefficients. Therefore, the case (ii) (when the test pattern is close enough to the training pattern p') is in fact the extreme situation of the case (i), and therefore also follows the PMRR method.

In a word, the memory recall process PMRR in PPNN testing stage is divided into the following steps: (a) Selecting the training pattern p' which is closest to the test pattern; (b) The previously stored memory in each of the parallel branches of hidden layers is retrieved individually first (Each of the P parallel branches of hidden layers generates its own test output O_{pj} separately within its own branch of hidden layers). (c) If the test pattern is not close enough to the training pattern p', the net test output is produced interpolatively and extrapolatively by means of weighted mean of each of all these individual test outputs from these parallel branches of hidden layers. (d) If the test pattern is close enough to a training pattern p', the net test output is simply the p'th parallel branch of hidden layers' test output itself.

5. SIMULATIONS

Because in current literatures, every researchers use different parameters and often did not mention the values of these parameters in their simulations, it is difficult to compare one's results directly with others. For the same reason as Tollenaere (1990), the simulations in this paper kept consistent within themselves, and did not compare directly with others results. We provide the values of parameters we used here in order for others to compare.

In order to contrast the effect produced by the topological difference only, we compared PPNN with BPNN under the conditions that all the other structures and parameters are the same for both of them except the topology.

We compared PPNN with BPNN for several problems to test their learnability and

generalization ability. We choose XOR problem, parity problem (Rumelhart, et al., 1986; Tollenaere, 1990), and the binary-to-local problem (Jacobs, 1988) to study the convergence rates and learnability because these problems are very difficult problems to learn, but are antigeneralization tasks (i.e. small difference in inputs will result in large difference in outputs). We choose the parabolic problem which maps the parabola function to study both the learning ability and the generalization ability because this problem is very difficult to learn and has infinite patterns to be used for testing.

Before going to the simulations, we want to make several points clear first:

(1). In order to apply the networks to more practical problems (with input and output values not restricted between 0 and 1) and in the meantime to meet the above requirements that BPNN and PPNN differs only in topology, both BPNN and PPNN in the following simulations had only one hidden layer with sigmoid activation function (Hornik, Stinchcombe, and White, 1990; Hornik, 1991). The threshold (bias) was treated as another weight with constant input 1. The output layer had linear activation function;

(2). The momentum strategy (Jacobs, 1988; Tollenaere, 1990) was used for both BPNN and PPNN;

(3). All the initial weights were generated randomly between -1 and 1;

(4). We used the mean absolute error (MAE) criterion to train both BPNN and PPNN. MAE criterion is a value set beforehand. If the mean absolute error (difference) between the desired outputs and the actual outputs of a network is below this MAE value, the training process is thought to be completed. Otherwise, we say that the network cannot converge (or fail to learn) at the preset MAE.

(5). Epoch is defined as the single sweep through all the training patterns followed by one update for every weight in BPNN (Tollenaere, 1990). However, for PPNN, because each pattern is trained within its own branch of hidden layers, its training is not accompanied by other training patterns. So there is no sweep through all training patterns in PPNN. We used training time to measure the local sweep (sweep within each branch) through each individual training pattern within its own branch of hidden layers in PPNN. Therefore, the average training time in PPNN corresponds to the epoch in BPNN.

(6). All the simulations were written in C and were run on VAX 8800.

5.1. XOR Problem

As mentioned above, in order to show the differences between the stereotopology and the planar topology, we kept BPNN and PPNN here the same in all aspects except the topology.

Both the learning rate and the smoothing factor were chosen as 0.5. The MAE criterion was 0.01.

The simulation result is shown in Table 1.

5.2. Parity Problem

We chose the size four parity problem (Rumelhart, Hinton, and Williams, 1986) with four inputs nodes, one hidden layer of four hidden nodes network for the parity problem. For the same reasons above, we kept BPNN differs from PPNN in topology only. So it was different from Rumelhart et al's treatment with fixed threshold and

nonlinear outputs. There were sixteen different patterns for training. Learning rate and smoothing factor were 0.5. The MAE criterion was 0.01.

The simulation result is shown in Table 1.

5.3. Binary-to-Local Problem

Binary-to-local problem is a good problem to study the learning ability of neural nets (Jacobs, 1988). The input vectors are binary values between zero and seven encoded by the three input nodes which take values 1 or 0. The desired output vectors are indicators of input values with local representation. Each of the eight output nodes is on only when the input values it represents is fed in.

In order to compare the topological effects, we used Jacobs' structure for both BPNN and PPNN: three input nodes; two hidden layers with one hidden node in the first hidden layer and eight hidden nodes in the second hidden layer; and eight output nodes. Both learning rate and smoothing factor were 0.5. MAE criterion was set to 0.01.

The simulation results is shown in Table 1.

5.4. Parabolic Problem

Parabolic problem is a task of mapping parabola function $y=x^2$. This is not a toy problem, but a difficult real problem (Leonard and Holz, 1988) because it is non-monotonous (it decreases when $x < 0$ and increases when $x > 0$), it has a multi-to-one mapping (both -x and x map the same y), and it has infinite points (coordinates pairs). Since there are infinite points in the domain and range of a parabola function, there are infinite patterns (points) available. Because the parabola curve has regularity, it is suitable for a trained net to infer (interpolate or extrapolate) unseen patterns based on the trained patterns. Therefore, the parabolic problem is a good problem to test the generalization ability of neural networks. So we used parabolic problem for both learning ability study (Table 1) and generalization ability study (Figure 4).

Due to the reasons discussed in (Xu and Zheng, 1992a), and because the parabolic problem has infinite patterns, we chose the one-to-one DR to implement the simulations for both BPNN and PPNN. The parabola here is an example of the n-dim static curve discussed in (Xu and Zheng, 1992a) with n=2. Because the x coordinate and y coordinate are in input space and output space of the neural nets respectively (to simulate the mapping from x space to y space), there is only one element (E=1) and one unit (U=1) for both input representation (a one-to-one DR) and output representation (a one-to-one DR too). The one input unit representing the x-coordinate of points on the parabola and the one output unit representing the y-coordinate of these points. The input unit takes analog values which are equal to the x-coordinate values of these points and the output unit takes analog values which are equal to the y-coordinate values of these points.

For example, for a training point (4, 16) on the parabola, $x=4$, $y=x^2=16$. The input unit then takes value 4 and the output unit takes value 16. Generally speaking, for any training point (x_o, y_o) on the parabola, the input unit takes the analog value x_o and the output unit takes the analog value y_o directly.

From above we see that the one-to-one DR is very natural (Xu and Zheng, 1992c) to us: each unit simply takes the analog values of the items (concrete coordinate

values here) contained within the elements (coordinates here).

(i). Learning ability study using parabolic problem.

With the one-to-one DR , we used one input node providing the input values of x and one output node giving the output values of y. The hidden layer of each branch in PPNN had 5 hidden nodes. In order to contrast the influence of hidden units number, the number of hidden units in BPNN was set to N and P∗N respectively. The learning rate and smoothing factor were 0.5. The MAE criterion was set to 0.01 for learning ability study in order to keep consistent with the other three problems in Table 1. Training patterns were 21 points uniformly distributed with equal distance 0.5 between each other in the domain [-5, 5].

The simulation results showed that BPNN failed to converge at the preset MAE criterion using the one-to-one DR . However, PPNN learnt the problem at the preset MAE criterion with great ease using the one-to-one DR. The conclusion here needs some further elucidation.

The reasons that we chose the one-to-one DR instead of the LR are as follows:

(1). There are infinite points (patterns) in the parabola problem. It is impossible to represent these infinite patterns with LR (Xu and Zheng, 1992b).

(2). Even for limited sampled point-set when the LR can be used, LR is less economic or less efficient than the one-to-one DR (Xu and Zheng, 1992b).

(3). Many quantities can be simply described by analog values. Although analog values can be transformed into binary ones, a big analog value may have to be represented by binary values with very large number of positions. It is not economic to transform analog values to binary values. For the spatial curve problem, because there are infinite points in the space, it's impossible to use binary representation to represent the infinite points. However, if we use the analog representations, all the infinite points can be easily represented by simply using the analog values of these points' coordinates. *What we need is the kind of representations that can use analog values directly to represent different quantities.* By taking analog values for the items contained with elements, one-to-one DR works in a way like this: it takes the analog values of the items occured in the real world to represent all possible items. So it works very well for applications with infinite patterns.

It is due to above reasons, we prefer the one-to-one DR to the LR for the applications with infinite patterns such as the parabola problem.

Because the performance of a neural network is determined not only by the learning algorithm, activation function, and network topology, but also by the (external) representations (Xu and Zheng, 1992a). In other words, the performance of a neural net is representation-sensitive. So we are very specific here in drawing our conclusions: BPNN fails when using the one-to-one DR, but PPNN works well when using the one-to-one DR. [We have used LR on BPNN for a set of sampled spatial points, and found that contrary to the one-to-one DR, LR works for BPNN (unpublished data).]

(ii). Generalization ability study using parabolic problem.

In generalization ability study, we trained both BPNN and PPNN with 101 points uniformly distributed with equal distance 0.1 between each other in the domain [-5, 5]. MAE criterion was set to 0.001 in order to achieve higher accuracy. PPNN converged

very quickly too at this higher MAE criterion with an average training time of 20, while BPNN failed to converge at the MAE criterion. BPNN finished its training at the preset maximum training times (epochs) 100,000 with a MAE value of 8.5 which was much higher than the MAE criterion 0.001 that was reached quickly (20 average times) by PPNN.

We tested the trained BPNN and PPNN with 201 points uniformly distributed with equal distance 0.05 between each other in the domain [-5, 5]. Among them, 101 points were seen in training by the nets, and the other 100 points were not seen before. The closeness criterion was 0.01. The actual test outputs are shown in Figure 4. It was shown that PPNN generalized better than BPNN.

6. DISCUSSION

6.1. Learnability

In PPNN, because the training pattern p (p = 1, 2, ..., P where P is the number of training patterns) has its own branch of hidden layers p, and this branch of hidden layers p has its own set of weights, the weights in the pth branch of hidden layers are updated by the pth training pattern only. All the other training patterns will not influence the weights in the pth parallel branch of hidden layers.

This is completely different from the situation in BPNN. In BPNN, because there is only one branch of hidden layers, all the training patterns share the same set of weights. These weights are updated by each training pattern according to back-propagation algorithm. Therefore the weights in BPNN are influenced by all the training patterns. There exists an interference process among all the training patterns. This interference leads to an oscillation process of weights values (This is not the oscillation of total MSE. In fact, the cost function value will decrease if the problem can be learned) and a compromise or trade-off process in BPNN. Each training pattern tends to update the weights to the values that best match its own input/output pair in order to minimize the cost function (total mean square error). Because all the training patterns share the same set of weights, the weights actually undergo an oscillation process from values that best fit one training pattern to values that best fit the next training pattern, and so on. The final weights values of the trained BPNN are actually the result of a compromise or trade-off process among all the training patterns. It is the process of reaching a compromise or trade-off that makes the training of BPNN very slow and brings about other drawbacks to BPNN. No matter what learning algorithm is used and/or what node responsive characteristics are assumed, this compromise requirement always exists as long as the network employs the BPNN's planar topology. The slow oscillation process to achieve a compromise or trade-off state is inherent to the BPNN's planar topology. So BPNN's planar topology sets limits for the improvements of the convergence rate and other performances of BPNN.

In PPNN, because the weights of the pth parallel branch of hidden layers is updated only by the training pattern p, these weights are updated straightforwardly to the optimum values that best match the input/output pair of the training pattern p. The weights in the pth parallel branch of hidden layers belong to the pth training pattern only, and are not shared by other training patterns. Therefore they are not affected by other training patterns. So there is no oscillation of weights, and no compromise or trade-off process here. Due to the lack of the weights oscillation and compromise

process, the local minima problem does not occur in PPNN, and the global minimum can be reached very quickly. Also because there is no compromise process, the global minimum can be reduced to any small value. So PPNN can converge quickly to problems with any preset convergence accuracy. Therefore, although we have not applied PPNN to *any* problems, we predict that PPNN can learn *any* problems with very fast convergence rate.

In a word, the PPNN (1) converges much faster than BPNN; (2) avoids the local minima; and (3) is able to converge to any preset accuracy, i. e. can learn any problems. So PPNN has overcome the three drawbacks of BPNN addressed in the Introduction part of this paper.

The parallel branches of hidden layers of the stereotopology ensures that PPNN can converge for all problems. The underlying mechanism for this property is that PPNN has a much higher memory accuracy and capacity than BPNN and other networks with planar topology. This comes from the non-interference distribution property of PPNN. The memory for each training pattern in PPNN is stored in its own branch of hidden layers. Since there is no interaction between these branches, there is no interference between the memories stored in these branches. This ensures both accuracy and high capacity of information storage in PPNN.

In sum, the *parallel* characteristic of the stereotopology ensures the topological speedup of PPNN and makes PPNN a network capable of learning any kinds of problems much faster than BPNN.

6.2. Generalization Ability

The test process is a generalization process, or a memory information retrieval process. PPNN's test process is very different from that of BPNN.

In BPNN, because all the training patterns share the same branch of hidden layers (there is only one branch of hidden layers in BPNN), memories of all the training patterns are stored in the same set of weights. In test, this sole branch of hidden layers is on and all the memories stored there are retrieved simultaneously with the same probability. This makes the BPNN an uniformly or equal-weighted distributed (instead of weighted distributed as that used in PPNN) interpolating and extrapolating generator.

In PPNN, the memory retrieval process is divided into two stages. First, the previously stored memories stored in each of the parallel branches of hidden layers are retrieved individually. Each of the P parallel branches of hidden layers generates its own test output independently within its own branch of hidden layers. Second, if the test pattern is not close enough to any training patterns, the net test output is produced interpolatively and extrapolatively by means of weighted mean of all the individual outputs from these parallel branches of hidden layers. If the test pattern is close enough to a training pattern p', the net test output is just the p'th parallel branch of hidden layers' test output. So it is the probabilistic characteristic of PPNN's memory retrieval rule that makes PPNN a better generator than BPNN (Figure 4).

From appearance, PPNN is similar to the lookup tables and nearest-neighbor algorithms. However, there are differences between them:

(1). PPNN is a network. The memories in it are stored by training/learning procedure instead of conventional programming method. This difference is very similar to the difference between the NETtalk and DECtalk (Anderson and Rosenfeld,

1988).

(2). PPNN's stereotopology brings similarities between the lookup tables, nearest-neighbor algorithms, and PPNN. However, PPNN has another feature in its testing stage: probabilistic memory retrieval rule (PMRR). It is PMRR that makes PPNN generalize better than BPNN instead of worse than it (Jacobs and Jordan, 1991).

6.3. Robustness

Robustness is used to describe a network's sensitivity to parameter values (Tollenaere, 1990). We used the XOR problem to study the robustness of PPNN for learning rate and smoothing factor.

For all the training patterns, PPNN was insensitive to the learning rate (when smoothing factor was 0.5) in the range from 0.3 to 7 (Figure 5), and was insensitive to smoothing factor (when learning rate was 0.5) in the range from 0 to 0.7 (Figure 6). Therefore PPNN is robust to both learning rate and smoothing factor in a wide range.

6.4. Binary Representations vs. Analog Representations

In (Xu and Zheng, 1992a), representations were classified according to the correspondence and location criteria. Among the representations classified there, some of them take binary values only (binary representations), and some of them may take analog values (analog representations). For the parabola problem in this paper, we discussed above that the analog representation one-to-one DR is superior to binary representations. Here we discuss this issue further from another viewpoint.

The relationship between the binary representation and the analog representation probably can be seen from the relationship between the conventional computing and the neural computing. In conventional computing, the basic computing element is the bit which can only take binary values. In neural computing, the basic computing element is the unit which can take analog values. The unit in neural computing can take binary values too. However, the true power of neural computing probably lies in the fact that its basic computing element -- unit -- can take analog values. Without using analog representations, the power of neural computing probably will not be completely employed.

7. CONCLUSION

Current BPNN has several problems which severely restrict its practical applications: (1) slow rate of convergence, (2) local minima, and (3) incapability of learning etc. PPNN was proposed to overcome these problems by adopting a new neural network topology----stereotopology----to replace the planar topology which most of the current neural networks are using. It was shown in this paper that (1) the *parallel* characteristic of PPNN's stereotopology makes PPNN a universal network capable of learning any kinds of problems with faster convergence rate than BPNN (thus PPNN has overcome the three problems associated with BPNN), and (2) the *probabilistic* characteristic of PPNN's memory retrieval rule PMRR ensures that PPNN has a better generalization ability than BPNN.

REFERENCES

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science,* **9,** 147-169.

Anderson, J. A., and Rosenfeld, E. (1988). *Neurocomputing: Foundations of Research.* Cambridge, MA: MIT Press.

Austin, J. (1988). Hybrid neural networks. *Neural Networks,* **1,** suppl. 1, 5.

Baba, N. (1989). A new approach for finding the global minimum of error function of neural networks. *Neural Networks,* **2,** 367-373.

Baldi, P., and Hornik, K. (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks,* **2,** 53-58.

Brent, R. P. (1991). Fast training algorithms for multilayer neural nets. *IEEE Transactions on Neural Networks,* **2,** 346-354.

Cuykendall, R., and Reese, R. (1988). Scaling techniques for neural algorithms. *Neural Networks,* **1,** Suppl. 1, 87.

Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks. CMU-CS-88-162.

Fogaca, M., Kramer, A., and Moore, B. (1988). Scalability issues in neural networks. *Neural Networks,* **1,** Suppl. 1, 92.

Hayduk, J. W., and Fong, Y. (1988). Fast learning of arbitrarily shaped decision regions in neural networks. *Neural Networks,* **1,** Suppl. 1, 25.

Hirose, Y., Yamashita, K., and Hijiya, S. (1991). Back-propagation algorithm which varies the number of hidden units. *Neural Networks,* **4,** 61-66.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks,* **4,** 251-257.

Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks,* **3,** 551-560.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks,* **1,** 295-307.

Jacobs, R.A., and Jordan, M.I. (1991). A competitive modular connectionist architecture. In: R.P. Lippmann, J.E. Moody, & D.S. Touretzky (eds), *Advances in Neural Information Processing Systems* 3, 767-773.

Le Cun, Y. (1985). A learning procedure for asymmetric threshold network. *Proceedings of Cognitiva,* **85,** 599-604.

Leonard, J., and Holz, M. (1988). System applications. In DARPA Neural Network Study, AFCEA.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE Acoustics Speech Signal Processing Magazine,* **4,** 4-22.

Madhavan, P. G., Xu, B., and Stephens, B. E., Low, W. (1991). On the convergence rate and generalization ability of tri-state neural networks. Proceedings of the Fourth Conference on Neural Networks and Parallel Distributed Processing, Ft Wayne.

Pao, Y., and Beer, R. D. (1988). The functional link net: a unifying network architecture incorporating higher order effects. *Neural Networks,* **1,** suppl. 1, 40.

Parker, D.B. (1985). Learning logic. Technical Report TR-47, MIT, Cambridge, MA.

Peterson, C., and Hartman, E. (1989). Explorations of the mean field theory learning algorithm. *Neural Networks,* **2,** 475-494.

Rigler, A. K., Irvine, J. M., and Vogl, T. P. (1991). Rescaling of variables in back propagation learning. *Neural Networks,* **4,** 225-229.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, (Editors), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Vol. 1, Foundations, Cambridge, MA: MIT Press.

Shawe-Taylor, J. S., and Cohen, D. A. (1990). Linear programming algorithm for neural networks. *Neural Networks,* **3,** 575-582.

Shepanski, J. F. (1988). Multilayer perceptron training using optimal estimation. *Neural Networks,* **1,** suppl. 1, 51.

Tollenaere, T. (1990). SuperSAB: Fast adaptive back propagation with good scaling properties. *Neural Networks,* **3,** 561-573.

Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T., and Alkon, D. L. (1988). Accelerating the convergence of the back-propagation method. *Biological Cybernetics,* **59,** 257-263.

Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. dissertation, Committee on Appl. Math., Harvard Univ., Cambridge, MA.

Winter, R., and Widrow, B. (1988). MADALINE Rule II: a training algorithm for neural networks. *Neural Networks,* **1,** suppl. 1, 148.

Xu, B., and Zheng, L. (1991). PPNN: A faster learning and better generalizing neural net. *Proceedings of IJCNN'91, Singapore.*

Xu, B., and Zheng, L. (1992a). On representations. *Proceedings of The SPIE Intelligent Information Systems OR'92--Science of Artificial Neural Networks,* Orlando, Florida.

Xu, B., and Zheng, L. (1992b). Capacity of representations. Submitted to *The Fourteenth Annual Conference of the Cognitive Science Society,* Bloomington, Indiana.

Xu, B., and Zheng, L. (1992c). Evaluations of representations. Submitted to *The Fourteenth Annual Conference of the Cognitive Science Society,* Bloomington, Indiana.

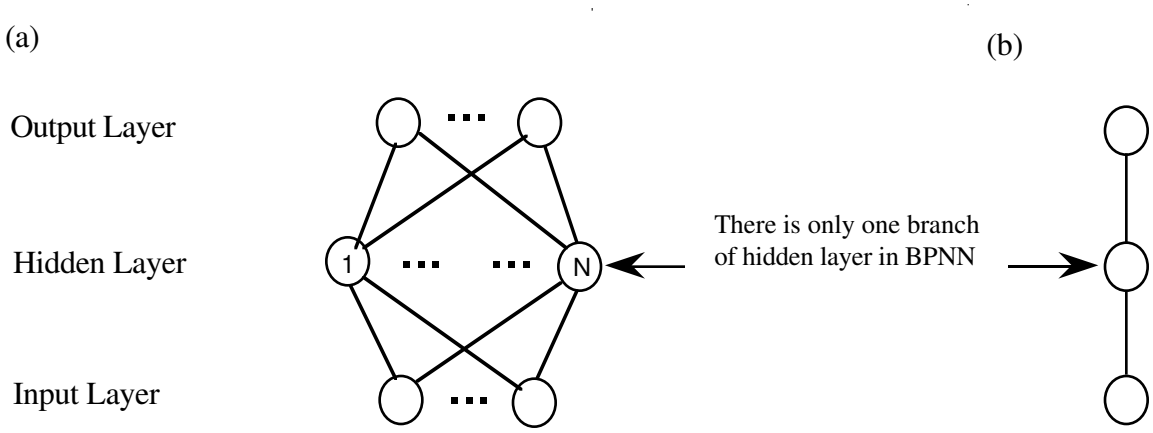
(a)
(b)
Output Layer
Hidden Layer
Input Layer
1
N
There is only one branch
of hidden layer in BPNN

Figure 1. The planar topology of BPNN. (a). The front view of BPNN planar topology. N is the number of hidden nodes. (b). The side view of BPNN planar topology. There is only one branch of hidden layers in BPNN. Therefore from side view, the input layer, the hidden layer, and the output layer can lie in one plane. So BPNN has a planar topology. Here only one hidden layer is shown. The conclusion is the same for nets with multi-hidden layers.

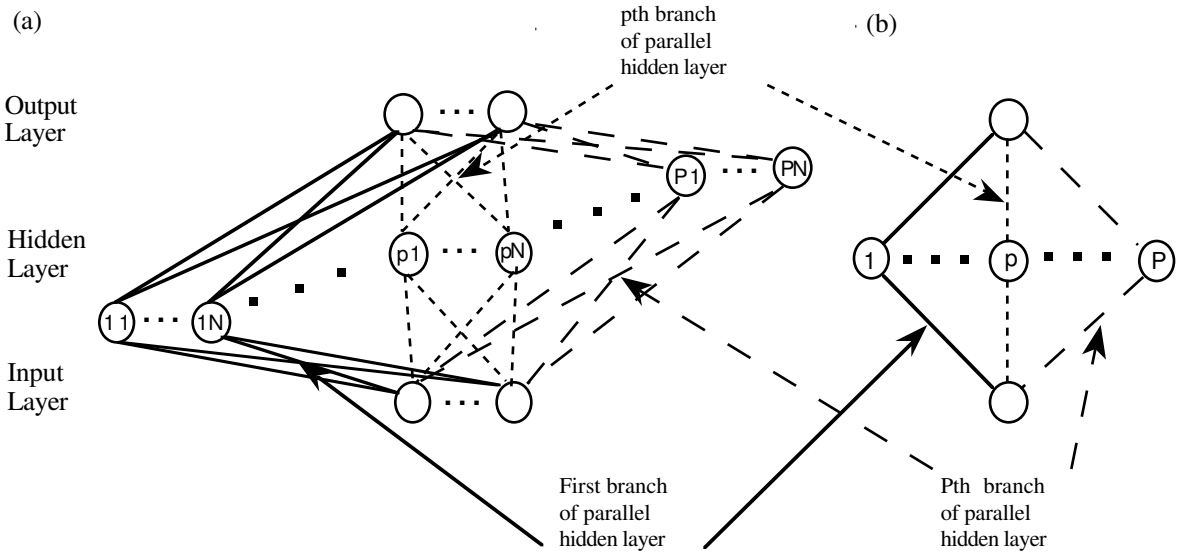
(a)
pth branch
of parallel
hidden layer
(b)
Output
Layer
Hidden
Layer
Input
Layer
11
1N
p1
pN
P1
PN
1
p
P
First branch
of parallel
hidden layer
Pth branch
of parallel
hidden layer

Figure 2. The stereotopology of PPNN. (a). The front view of PPNN stereotopology. N is the number of hidden nodes in each parallel branch of hidden layers. P is the number of training patterns which is equal to the number of parallel branches of hidden layers (p = 1, 2, ... , P). (b). The side view of PPNN stereotopology. There are P branches of parallel hidden layers in PPNN. From the side view, all these parallel branches of hidden layers can not lie in the same plane with the input layer and the output layer. So PPNN has a stereotopology instead of a planar one. Here in each parallel branch of hidden layers, only one hidden layer is shown. However, the conclusion is the same for nets with multi-hidden layers in each parallel branch of hidden layers.

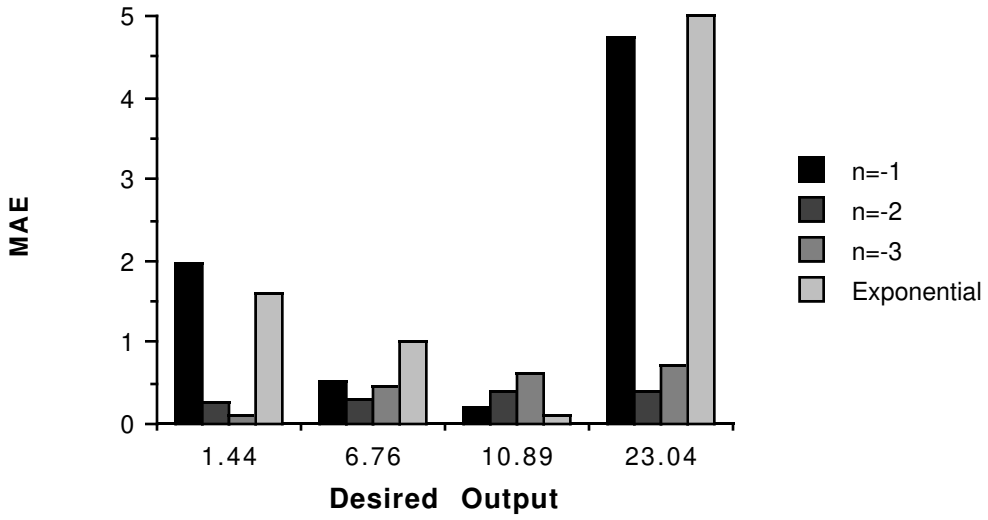
MAE
5
4
3
2
1
0
1.44
6.76
10.89
23.04
Desired Output
n=-1
n=-2
n=-3
Exponential

Figure 3. Comparison of different functions for the weighted mean's coefficients used in PPNN generalization process. Three power functions $a_p = \Delta_p^n$ (n= -1, -2, and -3) and one exponential function $a_p = \exp(-\Delta_p)$ were studied in PPNN tests for parabolic problem. The training patterns were eleven points uniformly distributed with distance 0.5 between each other in the domain [0,5]. Four test points (1.20, 1.44), (2.60, 6.76), (3.30, 10.89), and (4.80, 23.04) were arbitrarily selected to test which of the four functions gave the best generalization results. The mean absolute error (difference) MAE between the desired outputs and the actual outputs were drawn for every a_p functions at each test point. It was seen that the function $a_p = \Delta_p^{-2}$ gives the least overall MAE.

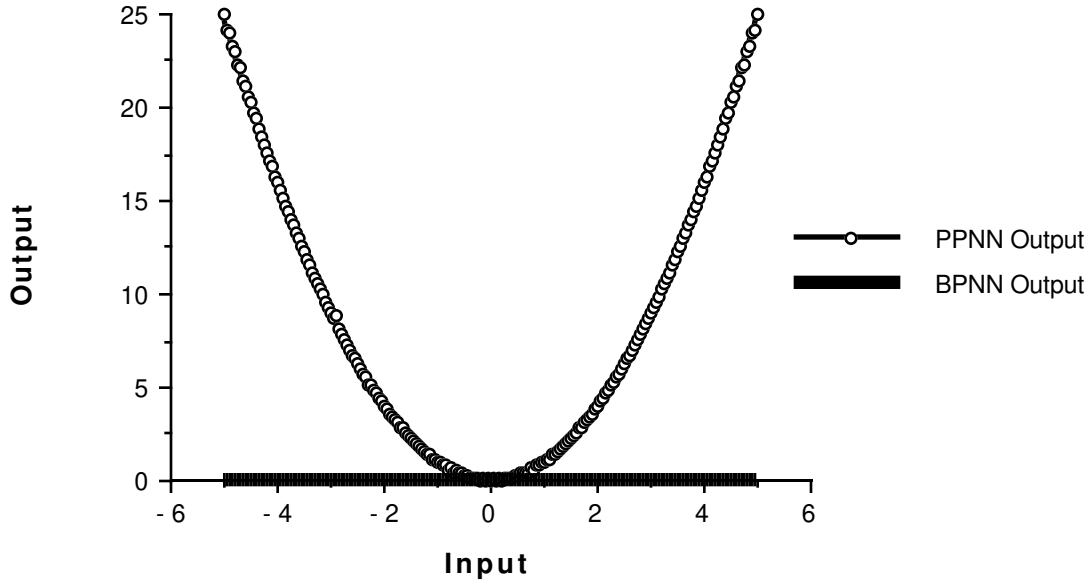
25
20
15
10
5
0
Output
- 6
- 4
- 2
0
2
4
6
Input
PPNN Output
BPNN Output

Figure 4. The net test outputs of BPNN and PPNN for parabolic problem. One-to-one DR was used for both the input representation and the output representation. Both BPNN and PPNN were trained with 101 patterns (points) uniformly distributed with a distance of 0.1 between each other in the domain [-5, 5]. MAE criterion was set to 0.001. The trained networks were tested with 201 points uniformly distributed with distance 0.05 between each other in the domain [-5, 5]. Among these test patterns, 101 were seen by the nets during training, and the other 100 patterns were not seen before. The closeness criterion was set to 0.01. The test outputs generated by PPNN for both the 101 seen patterns and the 100 unseen patterns were very close to the desired parabola curve $y=x^2$. However, the test outputs generated by BPNN for both the 101 seen and the 100 unseen patterns were lying on the straight line y=0 (on x-axis). Therefore, PPNN had a better generalization ability than BPNN when using the one-to-one DR.

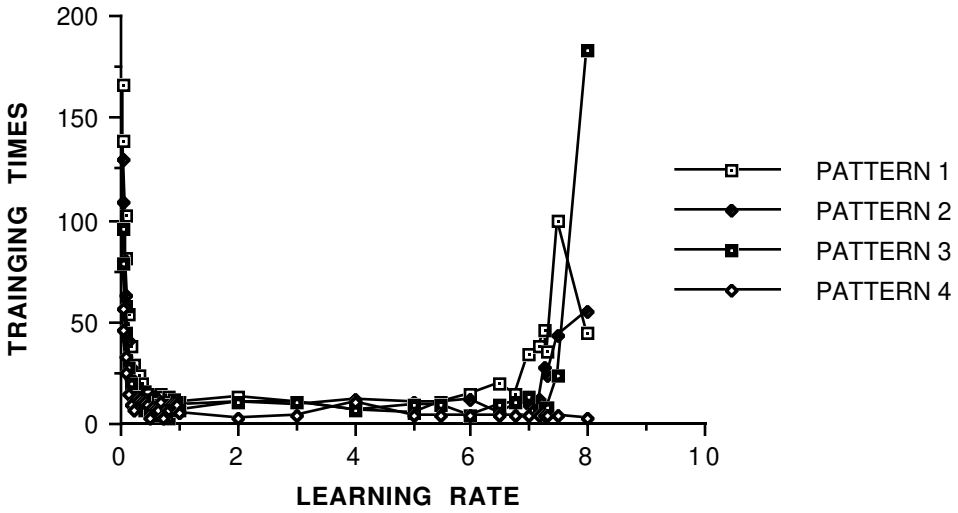
200
150
100
50
0
TRAINGING TIMES
0
2
4
6
8
10
LEARNING RATE
PATTERN 1
PATTERN 2
PATTERN 3
PATTERN 4

Figure 5. PPNN learning rate robustness. The XOR problem was used to study the robustness of learning rate. Smoothing factor was 0.5. MAE criterion was 0.01. Initial weights were random numbers in the range (-1, 1).

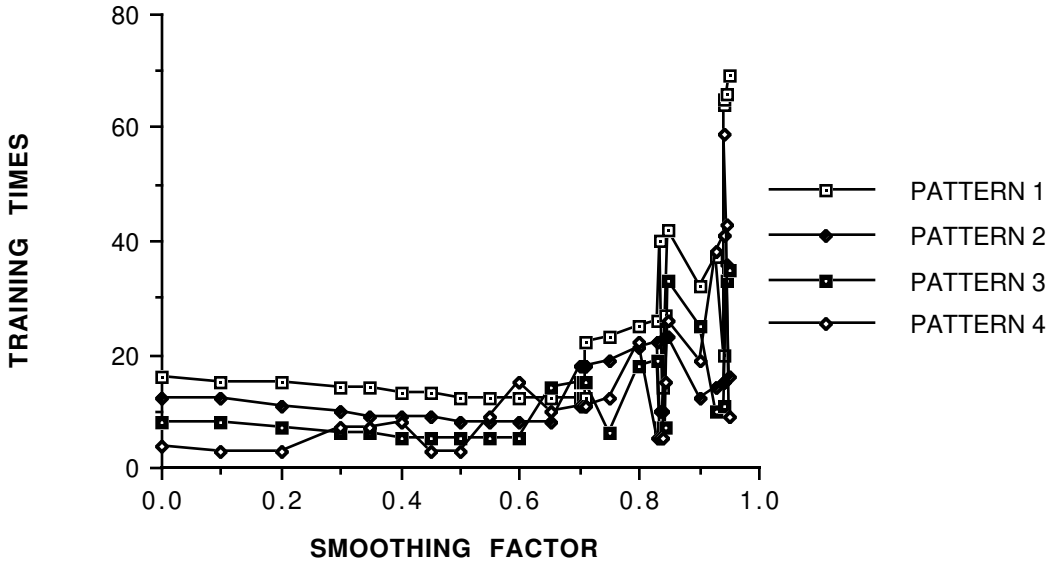
80
60
40
20
0
TRAINING TIMES
0.0
0.2
0.4
0.6
0.8
1.0
SMOOTHING FACTOR
PATTERN 1
PATTERN 2
PATTERN 3
PATTERN 4

Figure 6. PPNN smoothing factor robustness. XOR problem was used to study the robustness of smoothing factor. Learning rate was 0.5. MAE criterion was 0.01. Initial weights were in the range of (-1, 1).

Table 1. Comparison of Rate of Convergence between BPNN and PPNN

PROBLEMS			XOR	PARITY	BINARY-TO-LOCAL	PARABOLIC
BPNN[c]	$N_h=N$	MAE[a]	0.0089	0.0053	0.1437	8.5
		Training Times[b]	40490	100000	100000	100000
		Actual Training Time	11 min 33 sec	1 hr 18 min 15 sec	2 hr 14 min 55 sec	8 hr 21 min 36 sec
	$N_h=P*N$	MAE[a]	0.01	0.01	d	d
		Training Times[b]	260	3351	d	d
		Actual Training Time	4 sec	7 min 24 sec	d	d
PPNN		MAE[a]	0.01	0.01	0.01	0.01
		Training Times[b]	7	9	70	13
		Actual Training Time (Sec)	< 1	2	2	28

a. MAE criterion is a value set beforehand. If the mean absolute error (difference) between the desired outputs and the actual outputs of a network is below this value, the training process is thought to be completed at the accuracy of this MAE criterion. Otherwise, the training process has to be stopped by maximum training times limit. The MAE criterion for training here were set to 0.01 for all four problems. This criterion was very easily reached by PPNN for all four problems. However, for BPNN, only XOR problem reached this criterion; all the other three problems failed reaching it when the maximum training times 100,000 was reached. The MAE listed in BPNN for parity, binary-to-local and parabolic problems were the actual MAE values when the maximum training times limit 100,000 was reached.

b. The training times for BPNN were simply the epochs. PPNN has different training times for different training patterns trained within the corresponding branches of parallel hidden layers. The training times shown here for PPNN were the average training times over all these different branches of hidden layers for each problem.

c. Because PPNN contains a total of N∗P (N----hidden units number in each branch in PPNN; P----total number of parallel branches of hidden layers in PPNN) hidden units, we tested BPNN under two situations: (1) BPNN with a total of N hidden units

(this corresponds to the situation that BPNN overlaps all the branches of hidden layers in PPNN to one branch); (2) BPNN with a total of N∗P hidden units (this correspondes to the situation that BPNN has the same total hidden units as PPNN).

d. For the parabolic problem using BPNN, when N_h=N∗P=5x21=105, the MAE increased so rapidly that arithmetic fault (floating overflow) occured. The same phenomenon occured as long as N_h>30. For N_h=30, the MAE oscillated around 11.8. This showed that for BPNN, it's not necessary the more hidden units the better. It seemed that there was an optimum range of hidden units for BPNN.