A Multi-Sieving Neural Network Architecture That Decomposes Learning Tasks Automatically

BAO-LIANG LU, HAJIME KITA and YOSHIKAZU NISHIKAWA
Department of Electrical Engineering, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-01, Japan
E-mail: lbl@rotary.kuee.kyoto-u.ac.jp

Abstract—**This paper presents a multi-sieving network (MSN) architecture and a multi-sieving learning (MSL) algorithm for it. The basic idea behind MSN architecture is the multi-sieve method, that is, patterns are classified by a rough sieve at the beginning and done by finer ones gradually. MSN is constructed by adding a sieving module (SM) adaptively with progress of training. SM consists of two different neural networks and a simple logical circuit. MSL algorithm starts with a single SM, then does the following three phases repeatedly until all the training samples are successfully learned: (a) the learning phase in which the training samples are learned by the current SM, (b) the sieving phase in which the training samples that have been successfully learned are sifted out from the training set, and (c) the growing phase in which the current SM is frozen and a new SM is added in order to learn the remaining training samples. MSN architecture has several attractive properties such as automatic decomposition of learning tasks, modular structure, easy implementation of additional learning, overcoming a problem of local minima and fast convergence. The performance of MSN architecture is illustrated on two benchmark problems.**

I. INTRODUCTION

Back-propagation networks (BPNs) have been successfully applied to many pattern recognition problems. To date, a common feature of these successful applications is that either they are based on relatively small networks or they are simple classification problems. For more complex pattern classification problems, however, BPNs face many difficulties such as slow learning, trapping in local minima and necessity for selecting a suitable network size.

In order to overcome the above difficulties, several learning architecture for feedforward neural networks have been proposed, i.e., the *cascade correlation* architecture[1] and the *extentron algorithm*[2]. Fast convergence and powerful learning capability of these learning architectures have been reported. However, there are two main deficiencies shared by them: (a) it is difficult to modulize networks, and (b) all the parameters of the trained network must be changed when new samples are added.

In this paper, we propose a multi-sieving network (MSN) architecture and a multi-sieving learning (MSL) algorithm for it. MSN architecture can overcome the above deficiencies of the existing learning architectures. The basic idea behind MSN architecture is a problem solving method by human, i.e., the sieve method. Let's consider how humans are dealing with the classification problems. To illustrate the concept, we suppose a problem of seeking a very small grain of diamond from a huge pile of sands. The sizes of the sands are various and the sizes of a large part of the sands are greater than that of the diamond. Two problem solving methods can be described as follows:

(a). One seeks for the diamond in the huge pile of the sands directly checking whether an object is the diamond or not, piece by piece.

(b). One firstly uses various sizes of sieves from rough to fine to sift out the sands which are obviously greater than the diamond. With this method, the number of the sands will be reduced greatly. Then, one seeks for the diamond in a small pile of the sands efficiently.

Obviously, the second method is much more efficient than the first one. It is the second method that is adopted in our learning architecture.

II. NETWORK STRUCTURE

The multi-sieving network is illustrated in Fig. 1(a). It consists of several sieving modules connected in cascade. A sieving module may take one of two forms, i.e., RC-form or R-form as shown in Fig. 1(b), according to the learning task. The RC-form sieving module consists of a recognition network RN, a control network CN, an output judgement unit OJU, an AND gate, a NOT gate, and two logical switches. The R-form sieving module is similar to the RC-form, exclusive of the control network and the AND gate.

Before describing each element of the sieving module, we first introduce the output coding method used in the

0-7803-1901-X/94 $4.00 ©1994 IEEE

recognition network.

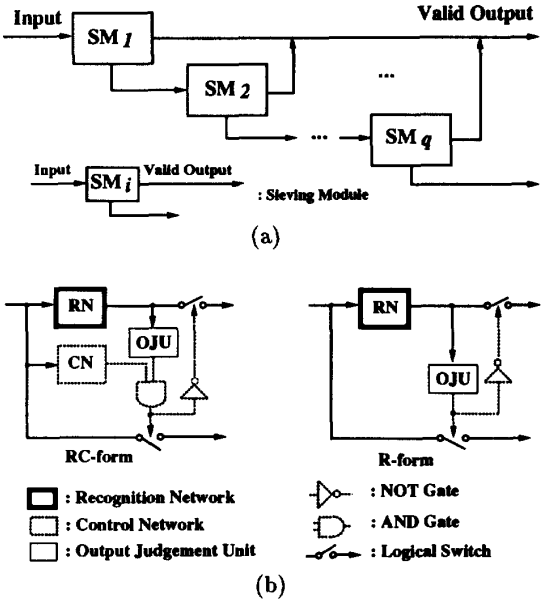

Figure 1: Structure of the multi-sieving neural network (a) and two forms of sieving modules (b).

A. Output Coding

We focus on classification tasks as an application of neural networks. That is, the network is required to divide input data into prescribed number of sets. We use the following output coding method for the recognition network: For classification task of $p+1$ sets, we use p or $p+1$ output units. For the kth recognition network RN_k, a desired output pattern $\bar{x}_i^{\mathcal{O}} = \{\bar{x}_{i1}^{\mathcal{O}}, \bar{x}_{i2}^{\mathcal{O}}, \cdots, \bar{x}_{iN_k}^{\mathcal{O}}\}$ must satisfies one of the following rules:

$$\begin{array}{ll} \text{(a)} & \forall j(\bar{x}_{ij}^{\mathcal{O}} < x_{\mathrm{low}}^{\mathcal{O}}) \text{ for } j \in B_k \quad \text{or} \\ \text{(b)} & \exists j(\bar{x}_{ij}^{\mathcal{O}} > x_{\mathrm{high}}^{\mathcal{O}}) \wedge \forall l_{\,l \neq j}(\bar{x}_{il}^{\mathcal{O}} < x_{\mathrm{low}}^{\mathcal{O}}) \\ & \text{for } j \text{ and } l \in B_k \end{array}$$

where $B_k = \{1, 2, \cdots, N_k\}$, N_k is the number of output units in RN_k, and $x_{\mathrm{low}}^{\mathcal{O}}$ and $x_{\mathrm{high}}^{\mathcal{O}}$ represent the low and high bounds for the outputs, respectively. For example, three output units can only represent four valid outputs as follows: $(0,0,0)$, $(0,0,1)$, $(0,1,0)$ and $(1,0,0)$. Other four codings, $(0,1,1)$, $(1,0,1)$, $(1,1,0)$ and $(1,1,1)$, are considered to be invalid.

For a given input pattern $\bar{x}_i^{\mathcal{I}}$, RN_k may generate three kinds of actual outputs:

(a). *Valid output* : The valid output is the correct output generated by the recognition network. That is,

$$\forall j \mid \bar{x}_{ij}^{\mathcal{O}} - x_{ij}^{\mathrm{RN}_k} \mid < \delta \quad \text{for } j \in B_k \tag{1}$$

where $\bar{x}_{ij}^{\mathcal{O}}$ is the desired output of the jth unit, $x_{ij}^{\mathrm{RN}_k}$ is the actual output of the jth output unit of RN_k, and δ denotes a tolerance. If the desired values of output units are set to 0 and 1, then, $x_{\mathrm{low}}^{\mathcal{O}} = \delta$ and $x_{\mathrm{high}}^{\mathcal{O}} = 1 - \delta$.

(b). *Pseudo valid output*: The recognition network may generate an output which follows the coding rule mentioned above, but is not correct. We called such an output a *pseudo valid output*:

$$\begin{array}{l} \exists j_{\,j \neq h}(x_{ij}^{\mathrm{RN}_k} > x_{\mathrm{high}}^{\mathcal{O}}) \wedge \forall l_{\,l \neq j}(x_{il}^{\mathrm{RN}_k} < x_{\mathrm{low}}^{\mathcal{O}}) \ \vee \\ \forall l(x_{il}^{\mathrm{RN}_k} < x_{\mathrm{low}}^{\mathcal{O}}) \quad \text{for } j \text{ and } l \in, B_k \end{array} \tag{2}$$

where the desired output of the hth unit satisfies $\bar{x}_{ih}^{\mathcal{O}} > x_{\mathrm{high}}^{\mathcal{O}}$.

(c). *Invalid output:* Otherwise.

For example, if the desired output pattern is (0, 0, 1), $\delta = 0.2$, $x_{\mathrm{high}}^{\mathcal{O}} = 0.8$, and $x_{\mathrm{low}}^{\mathcal{O}} = 0.2$, then, (0.1, 0.1, 0.9) is a valid output, (0.9, 0.1, 0.1) and (0.1, 0.1, 0.1) are two pseudo valid outputs, and (0.9, 0.1, 0.9) is a invalid output.

B. Output Control

In order to differentiate among valid, pseudo valid and invalid outputs, the outputs produced by the recognition network are classified and controlled by an output control circuit as drawn in dotted line in Fig. 1(b). The output control circuit consists of an output judgement unit OJU, a control network CN, an AND gate, a NOT gate , and two logical switches.

(a) The output judgement unit is used to differentiate the invalid output from other two kinds of outputs. OJU generates 1 or 0 according to

$$\mathcal{O}_{\mathrm{OJU},k} = \begin{cases} 1, & \text{if } x_{ij}^{\mathrm{RN}_k} \text{ is a valid or pseudo valid output ;} \\ 0, & \text{Otherwise} \end{cases} \tag{3}$$

where $\mathcal{O}_{\mathrm{OJU},k}$ is the output of OJU in the kth sieving module, $A = \{1, 2, \cdots, t_k\}$, t_k is the number of examples used for training the kth recognition network.

(b) The control network is used to differentiate the valid output from the pseudo valid output. Its output is also 1 or 0, which is determined by

$$\mathcal{O}_{\mathrm{CN},k} = \begin{cases} 1, & \text{if } x_{ij}^{\mathrm{RN}_k} \text{ is a valid output;} \\ 0, & \text{Otherwise} \end{cases} \tag{4}$$

where $\mathcal{O}_{\mathrm{CN},k}$ is the output of the control network in the kth sieving module.

(c) The logical switch works as follows: If the input is "1", then the data are blocked by it. Otherwise, the data pass through it.

III. Multi-Sieving Learning Algorithm

Let T_1 be a set of t_1 training samples to be learned:

$$T_1 = \{(\bar{x}_i^{\mathcal{I}}, \bar{x}_i^{\mathcal{O}}) \mid \text{for } i = 1, 2, \cdots, t_1\} \tag{5}$$

where $\bar{x}_i^{\mathcal{I}} \in \mathbf{R}^{N_{\mathcal{I}}}$ and $\bar{x}_i^{\mathcal{O}} \in \mathbf{R}^{N_{\mathcal{O}}}$ are the input and the desired output of the ith sample, respectively.

The multi-sieving learning algorithm works as follows:

Step 1: Initially, a recognition network, namely RN_1, is trained on the original set T_1. Let $m = 1$, and proceed to the following steps.

Step 2: Compute the number of valid outputs, $N_{\mathrm{vo},m}$, and the number of pseudo valid outputs, $N_{\mathrm{pvo},m}$, according to Eqs. (1) and (2), respectively. If $\sum_{i=1}^{m} N_{\mathrm{vo},i} = t_1$, i.e., all t_1 samples are learned by the multi-sieving network, then the training is completed. If $N_{\mathrm{pvo},m} > 0$, i.e., if there are $N_{\mathrm{pvo},m}$ pseudo valid outputs, then, a control network, namely CN_m, is trained on the set of $N_{\mathrm{vo},m} + N_{\mathrm{pvo},m}$ samples, corresponding to valid or pseudo valid outputs generated by RN_m. If $N_{\mathrm{pvo},m} = 0$, i.e., if there are no pseudo valid outputs, then the control network is unnecessary.

Step 3: Freeze the parameters of RN_m and CN_m (if it exists), and construct the mth sieving module as shown in Fig. 1(b).

Step 4: Remove $N_{\mathrm{vo},m}$ samples which have been successfully classified by RN_m from T_m and create a new set of t_{m+1} ($t_{m+1} = t_m - N_{\mathrm{vo},m}$) samples T_{m+1}, which are not classified by RN_m. Let $m = m + 1$ and go back to the step 2.

It should be noted that we assume that the control network CN_m always learns the classification of the valid and the pseudo valid outputs successfully.

The multi-sieving learning algorithm has several attractive properties such as:

(a) Once a sample is learned by a sieving module, then this sample will be removed from the set of samples and never been used in the training process. As a result, the deeper the multi-sieving network is, the fewer the samples become.

(b) A complex task can be decomposed into several manageable subtasks automatically, and each subtask can be learned by a sieving module.

(c) By adjusting the sizes of the recognition networks and the epochs used for training them, the number of sieving modules can be controlled.

(d) Various sorts of the structures of RN_m and CN_m and the training algorithms for them can be chosen by the user.

(e) The network obtained by MSL algorithm is modulized. Thus, it can be constructed and implemented easily in hardware.

IV. Implementation of Additional Learning

In many applications of neural networks, the trained network may not have good generalization capability since the number of training data picked up from environments is limited. Consequently, after the training, we must examine the generalization capability of the network. If the network has poor generalization capability, we should retrain the network by adding new data. In such a case, the additional learning is required. In this section, we present an algorithm for implementing additional learning based on MSN architecture and MSL algorithm.

Suppose a set of t_1 samples T_1 has been successfully learned by an MSN with q sieving modules, namely $\mathrm{MSN}_{\mathrm{old}}^q$. Now, the problem is how to add a new set of u_1 samples U_1

$$U_1 = \{(\bar{x}_j^{\mathcal{I}}, \bar{x}_j^{\mathcal{O}}) \mid \text{for } j = 1, 2, \cdots, u_1\} \tag{6}$$

to $\mathrm{MSN}_{\mathrm{old}}^q$. Without loss of generality and for simplicity of illustration, we also suppose that (a) all the q sieving modules are RC-form sieving modules and (b) the numbers of the valid outputs and the pseudo valid outputs produced by the kth recognition network RN_k^q in $\mathrm{MSN}_{\mathrm{old}}^q$ are $N_{\mathrm{vo},k}^q$ and $N_{\mathrm{pvo},k}^q$, respectively. Let $N_{\mathrm{vo},k}^{\mathrm{new}}$ and $N_{\mathrm{pvo},k}^{\mathrm{new}}$ be the numbers of valid outputs and pseudo valid outputs generalized by the kth recognition network RN_k^q, respectively, when the new training inputs are presented to $\mathrm{MSN}_{\mathrm{old}}^q$.

Based on MSN architecture and MSL algorithm, the additional learning can be implemented as follows:

Step 1: Initially, present all u_1 new training inputs to the first recognition network RN_1^q, let $m = 1$, and do the following steps.

Step 2: Compute $N_{\mathrm{vo},m}^{\mathrm{new}}$ and $N_{\mathrm{pvo},m}^{\mathrm{new}}$ according to Eqs. (1) and (2).

Step 3: If $((m \le q) \wedge (\sum_{i=1}^{m} N_{\mathrm{vo},i}^{\mathrm{new}} = u_1))$, i.e., all u_1 new training inputs are generalized correctly by RN_1^q through RN_m^q, then the additional learning is completed.

Step 4: If $((m > q) \wedge (\sum_{i=1}^{q} N_{\mathrm{vo},i}^{\mathrm{new}} < u_1))$, then go to step 5. If $((m \le q) \wedge (\sum_{i=1}^{m} N_{\mathrm{vo},i}^{\mathrm{new}} < u_1))$, then replace CN_m^q with a new control network $\mathrm{CN}_m^{\mathrm{new}}$ which is trained on a set of $N_{\mathrm{vo},m}^{\mathrm{new}} + N_{\mathrm{vo},m}^q + N_{\mathrm{pvo},m}^{\mathrm{new}} + N_{\mathrm{pvo},m}^q$ training data. If $((m \le q) \wedge (\sum_{i=1}^{m} N_{\mathrm{vo},i}^{\mathrm{new}} < u_1) \wedge (N_{\mathrm{vo},m}^{\mathrm{new}} = 0) \wedge (N_{\mathrm{pvo},m}^{\mathrm{new}} = 0))$, then CN_m^q remains unchanged. Let $m = m + 1$, and go back to step 2.

Step 5: Remove $\sum_{i=1}^{q} N_{\mathrm{vo},i}^{\mathrm{new}}$ samples, which are generalized correctly by RN_1^q through RN_q^q, from U_1 and train a new multi-sieving network, namely $\mathrm{MSN}_{\mathrm{new}}^p$, on a set of $u_1 - \sum_{i=1}^{q} N_{\mathrm{vo},i}^{\mathrm{new}}$ samples according to MSL algorithm.

Step 6: Suppose the set of $u_1 - \sum_{i=1}^{q} N_{\mathrm{vo},i}^{\mathrm{new}}$ samples has

been successfully learned by MSN^{p}_{new}. Connect MSN^{q}_{old} to MSN^{p}_{new} in series as shown in Fig. 2.

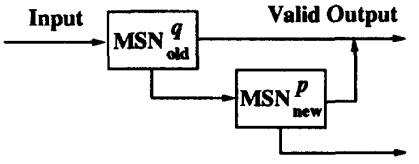

Figure 2: The connection of MSN^{q}_{old} to MSN^{p}_{new}.

V. Simulation Results

In this section, we will describe simulations to study the performance of MSNs in comparison with that of BPNs. The structure of RN_m and CN_m and the training algorithm for them are chosen to be multilayer quadratic perceptron (MLQP)[3] and the back-propagation algorithm[4], respectively. MLQP is an extension of multilayer perceptron. In MLQP, quadratic terms are introduced into the net input as well as linear terms in the conventional multilayer perceptron. It has been shown that MLQP is far superior to the conventional BPN in convergence for pattern classification problems[3]. In the simulations, the learning rates are experimentally optimized in convergence speed for the specified problems with a constant coefficient (0.9) for the momentums.

A. Example 1

The "two-spirals" problem is chosen as a benchmark for this study because it is an extremely hard problem for BPNs[5]. The training set consists of 194 $\langle x, y \rangle$ points at which the network should output 0's or 1's as shown in Fig. 3(d).

This problem is learned by a multi-sieving network with three sieving modules. The recognition network in each sieving module has 2 input, 5 hidden and 1 output units. All the sieving modules take the R-form as shown in Fig. 1(b) because there are no pseudo valid outputs in each sieving stage. In each sieving stage, the learning is stopped after 10,000 epochs if the total error is greater than a given value. Figs. 3(a) through 3(c) show the patterns that are classified by the first through the third sieving modules. The response plots of the first through the third sieving modules and the whole MSN are illustrated in Figs. 4(a) through 4(d). The CPU time for training all the recognition networks in three R-form sieving modules is about 2075 seconds on Sparc ELC workstation.

This problem is also learned by a MLQP with 2 input, 40 hidden and 1 output units. The network is trained by the back-propagation algorithm. After 500,000 epochs, the network has not yet achieved the desired total error. The learning curve and the response plot of the network are shown in Figs. 5(a) and 5(b), respectively. The CPU time for training this network is about 160,000 seconds. Comparing the learning results shown in Figs. 4(d) and 5(b) and the CPU time, we see that the multi-sieving network is much better than MLQP.

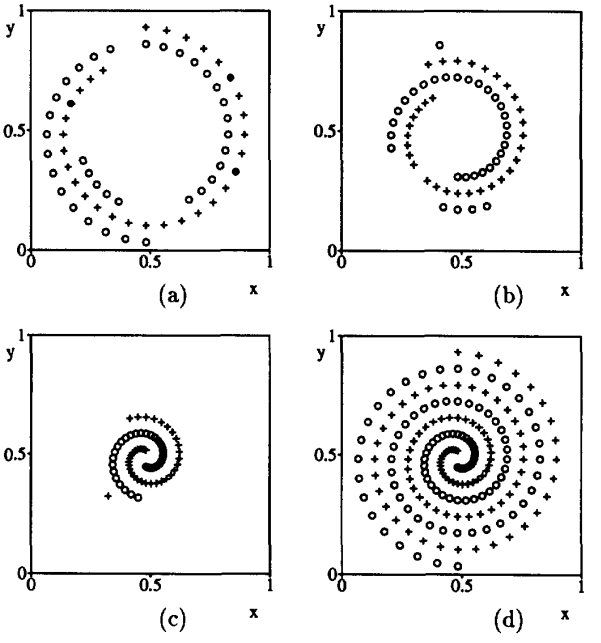

Figure 3: Patterns classified by the first sieving module (a), the second sieving module (b), the third sieving module (c), and the whole multi-sieving network (d). For symbols "o" and "+", the network is required to generate output 0 and 1, respectively.

B. Example 2

In this example, we will demonstrate how to implement additional learning by use of MSN architecture and MSL algorithm. A simplified version of the "two-spirals" problem shown in Fig. 6(a) is learned by a single recognition network RN_1. The response plot of RN_1 is illustrated in Fig. 7(a). Now, we add 16 new samples as shown in Fig. 6(b). We construct a MSN by adding another sieving module as shown in Fig. 8 to learn the augmented problem. The role of CN_1 is to differentiate the 16 new samples from the original samples. RN_2 is used to recognize the 16 new samples. After the 16 new samples are learned by RN_2, the response plot of the whole MSN is illustrated in Fig. 7(d). Figures 7(b) and 7(c) show the response plots of CN_1 and RN_2, respectively. From Figs. 7(a) and 7(d), we see that the 16 new samples have been added to the network without destroying any parameters of RN_1.

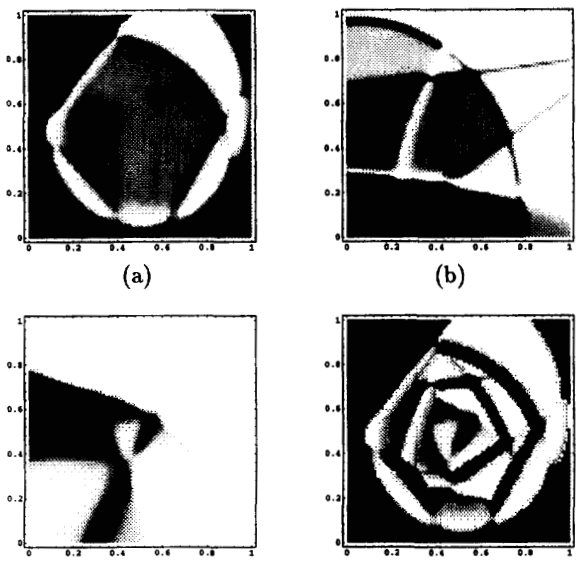

Figure 4: Response plots of the first sieving module (a), the second sieving module (b), the third sieving module (c), and the whole multi-sieving network (d). Black and white represent output of "0" and "1", respectively, and grey represents intermediate value.

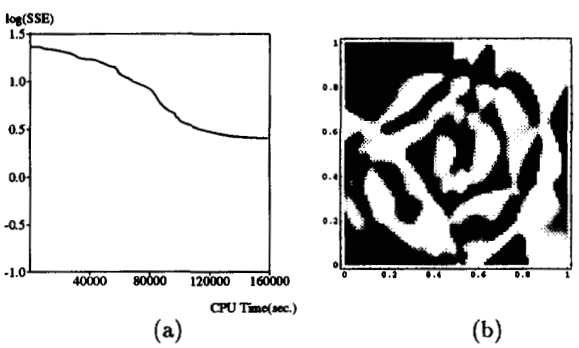

Figure 5: The learning curve (a) and the response plot of MLQP for the "two-spirals" problem (b).

C. Example 3

This example demonstrates how to overcome the problem of local minima by use of MSN architecture. We perform simulation with a speaker independent vowel recognition problem [6]. The vowel data consisting of 528 training data and 462 test data are taken from *CMU learning benchmark database* [7].

Initially, a single recognition network with 10 input, 11 hidden, and 11 output units, namely RN_1, is trained on a set of 528 samples by the backpropagation algorithm [4]. After 500,000 epochs of training, the total error is still about 3.21. The learning curve is illustrated in Fig. 9(a). From this figure, it seems that the network may be trapped in a local minima, and the network can not converge to the desired error, which is set to 0.01 in this simulation, without changing the initial parameters or network size.

Examining 528 training data, we see that only 6 data are not correctly classified by RN_1, and other 522 data have been successfully learned. There are no pseudo valid outputs formed in this case. In order to overcome the local minima and achieve the desired total error, we construct MSN by adding a sieving module to RN_1. The first sieving module is chosen to be R-form since $N_{vo,1} = 0$. A recognition network with 10 input, 3 hidden, and 11 output units, namely RN_2, is selected to learn the remaining 6 data which are not classified by RN_1. The learning curve is shown in Fig. 9(b). From this figure, we see that RN_2 converges to the desired error quickly. After the training, the network obtained by the MSL algorithm is a MSN with two R-form sieving modules. Examining the generalization capability of RN_1 amd MSN with two sieving modules on 462 test data, we obtain 38.74% correct rate for RN_1, and 41.13% for MSN[1]. From the above results, we see that the MSN architecture can overcome local minima efficiently and the generalization capability is improved slightly after the local minima is overcome.

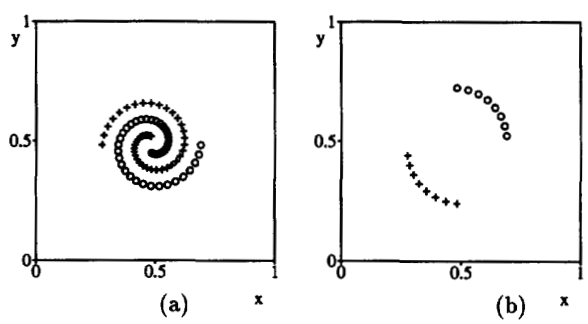

Figure 6: The original 82 samples (a) and the 16 new samples (b).

IV. Conclusion

In this paper, a new neural network architecture MSN and a learning algorithm MSL for it are proposed. MSN architecture has several advantages over BPNs. The most important advantages are automatic decomposition of learning tasks and easy implementation of additional learning. The simulation results show that MSN architecture overcomes the difficulties of local minima and slow convergence, which are encountered in BPNs, by decomposing learning tasks automatically.

[1] For this test data, it is difficult to obtain a high correct rate. Robinson [6] has reported that a correct rate about 44% is obtained by a multilayer perceptron with 11 hidden units.

Further refinement of MSN architecture with large pattern classification application is a subject to be studied in the future work.

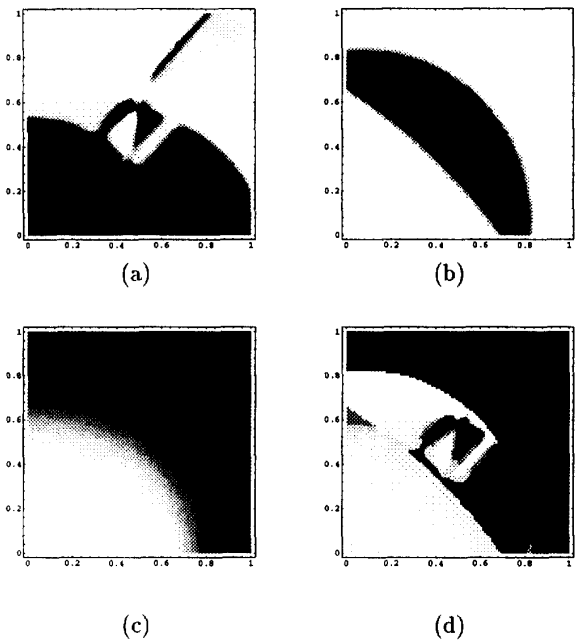

Figure 7: Response plots of RN_1 (a), CN_1 (b), RN_2 (c), and MSN (d).

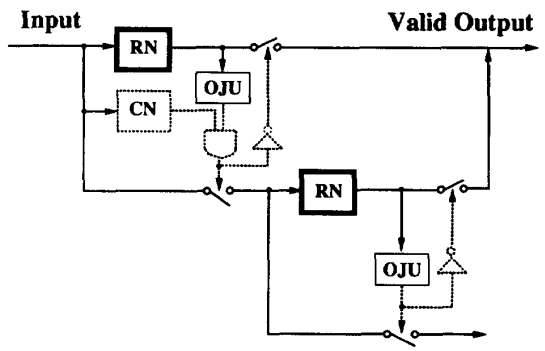

Figure 8: MSN for implementing the additional learning.

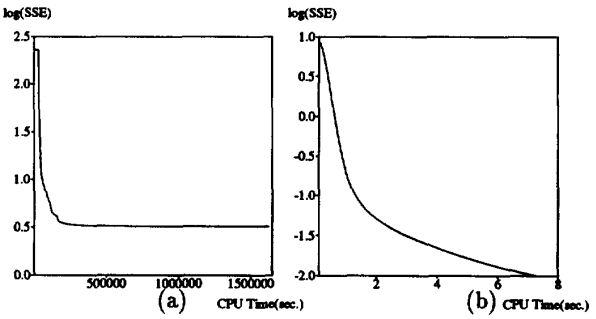

Figure 9: The learning curves of RN_1 (a) and RN_2 (b).

References

[1] Fahlman, S. and Lebiere, C. : "The cascade-correlation learning architecture", *Carnegie-Mellon University Report CMU-CS-90-100*, 1990.

[2] Baffers, P. T. and Zelle, J. M. : "Growing layers of perceptrons: Introducing the extentron algorithm", *Proc. of International Joint Conference on Neural Networks*, pp. II-392-II-397, Baltimore, MD, 1992.

[3] Lu, B. L., Bai, Y., Kita, H. and Nishikawa, Y. : "An Efficient multilayer quadratic perceptron for pattern classification and function approximation", *Proc. of the International Joint Conference on Neural Networks'93*, vol. II, pp. 1385-1388, Nagoya, 1993.

[4] Oogen, A. V. and Nienhuis, B. : "Improving the convergence of the backpropagation algorithm", *Neural Networks*, vol. 5, pp. 465-471, 1992.

[5] Lang, K. and Witbrock, M. : "Learning to tell two spirals apart", *Proc. of 1988 Connectionist Models Summer School*, Morgan Kanfann, pp. 52-59, 1988.

[6] Robinson, A. J. : "Dynamic error propagation networks", *Cambridge University, PhD. Thesis*, 1989.

[7] White, M.: "CMU learning benchmark database", *Carnegie Mellon University, School of Computer Science*, 1993.