A Parallel and Modular Multi-Sieving Neural Network Architecture with Multiple Control Networks

BAO-LIANG LU[1] and KOJI ITO[2]
[1]Bio-Mimetic Control Research Center,
The Institute of Physical and Chemical Research (RIKEN)
3-8-31 Rokuban, Atsuta-ku, Nagoya 456, Japan
[2]Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology
4259, Nagatsuda, Midori-ku, Yokohama 226, Japan
lbl@nagoya.riken.go.jp ito@ssie.titech.ac.jp

Abstract

We have proposed a constructive learning method called multi-sieving learning for implementing automatic decomposition of learning tasks and a parallel and modular multi-sieving network architecture in our previous work. In this paper we present a new parallel and modular multi-sieving neural network architecture to which multiple control networks are introduced. In this architecture the learning task for a control network is decomposed into a finite set of manageable subtasks, and each subtask is learned by an individual control sub-network. An important advantage of this architecture is that the learning tasks for control networks can be learn efficiently, and therefore automatic decomposition of complex learning tasks can be achieved easily.

1 Introduction

Some of the most important problems in neural networks, which hinder the progress of neural network methods for dealing with large and complex learning tasks, may be enumerated as follows: (a) The network architecture is monolithic; (b) The learning algorithms cannot decompose complex learning tasks automatically into relatively simple subtasks that can be learned by relatively small subnetworks; (c) Even few simple modifications to learning tasks are to be carried out, all the parameters of trained networks must be adjusted. Although several constructive *or* modular neural network architectures have been proposed such as tiling algorithm [9], the cascade correlation architecture [2], and hierarchies of adaptive experts [3], some of the problems mentioned above are still remained.

For tackling the above problems, we have proposed a constructive learning method called multi-sieving learning (MSL) and a parallel and modular multi-sieving network architecture (PMSN) in our previous work [6, 7, 8]. The basic idea behind MSL is the multi-sieving method. Patterns are classified by a rough sieve at the beginning and re-classified further by finer ones in subsequent stages. The MSL algorithm starts with a single sieving module (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. PMSNs are constructed by adding a SM adaptively with progress of learning.

In PMSN, the assumption is made that the control network CN_i in the ith sieving module always learns the classification of the *valid* and *pseudo valid* outputs produced by the ith recognition network RN_i successfully. Clearly, this is a strong assumption. This assumption becomes a bottleneck problem of multi-sieving learning because decomposition of learning tasks can not go forward further if the control network can not converge.

In this paper we present a new parallel and modular multi-sieving neural network architecture to which multiple control networks are introduced. In this architecture the learning task for a control network is decomposed into a finite set of manageable subtasks according to the numbers of the valid and pseudo valid outputs produced by a related recognition network, and each subtask is learned by an individual control sub-network. An important advantage of this architecture is that the tasks for control networks can be learn efficiently, and therefore automatic decomposition of complex learning tasks can be achieved easily.

2 PMSN-MC Architecture

2.1 Sieving Modules

The block diagram of a trained PMSN-MC is illustrated in Fig.1, which is the same as that of PMSN. All the sieving modules (SMs) in PMSN-MC are connected in parallel. The ith sieving module SM_i in PMSN-MC may take one of the two forms according to the actual outputs produced by the ith recognition network RN_i, i.e., RC-form or R-form as shown in Figs. 2(a) and 2(b), respectively.

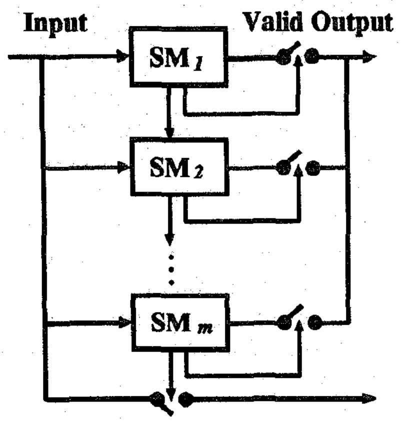


Figure 1: The block diagram of a trained PMSN-MC.

2.2 Output Representation Scheme

Various output representation schemes can be used to represent training outputs in recognition networks, such as binary coding, Gray coding, and 1-out-of-N coding. In this paper we use a modified 1-out-of-N coding method for recognition networks. For $p+1$ classes of training output patterns, we use p output units. For RN_i, the kth desired output pattern $\bar{x}^{\mathcal{O}}_k = \{\bar{x}^{\mathcal{O}}_{k1}, \bar{x}^{\mathcal{O}}_{k2}, \cdots, \bar{x}^{\mathcal{O}}_{kN_i}\}$ must satisfy one of the following rules:

$$\forall j(\bar{x}^{\mathcal{O}}_{kj} \leq x^{\mathcal{O}}_{\text{low}}) \tag{1}$$

$$\exists j(\bar{x}^{\mathcal{O}}_{kj} \geq x^{\mathcal{O}}_{\text{high}}) \wedge \forall l_{\,l \neq j}(\bar{x}^{\mathcal{O}}_{kl} \leq x^{\mathcal{O}}_{\text{low}}) \quad \text{for } j \text{ and } l \in B_i \tag{2}$$

where $B_i = \{1, 2, \cdots, N_i\}$, N_i is the number of output units in RN_i, and $x^{\mathcal{O}}_{\text{low}}$ and $x^{\mathcal{O}}_{\text{high}}$ represent the low and high bounds for the outputs, respectively. For example, three binary 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 codes, $(0,1,1)$, $(1,0,1)$, $(1,1,0)$ and $(1,1,1)$, are considered to be invalid.

2.3 Actual Outputs

For the kth input pattern $\bar{x}^{\mathcal{I}}_k$, RN_i may generate three kinds of actual outputs:

(a). *Valid output*: The valid output is a correct output and satisfies

$$\forall j \mid \bar{x}^{\mathcal{O}}_{kj} - x^{\mathrm{RN}_i}_{kj} \mid \leq \delta \quad \text{for } j \in B_i, \tag{3}$$

where $\bar{x}^{\mathcal{O}}_{kj}$ is the desired output of the jth unit, $x^{\mathrm{RN}_i}_{kj}$ is the actual output of the jth output unit of RN_i, and δ denotes a tolerance. If the desired values of output units are set to 0 or 1, then, $x^{\mathcal{O}}_{\text{low}} = \delta$ and $x^{\mathcal{O}}_{\text{high}} = 1 - \delta$. If $x^{\mathrm{RN}_i}_k$ is a valid output, this means that the kth training data, $(\bar{x}^{\mathcal{I}}_k, \bar{x}^{\mathcal{O}}_k)$, has been learned properly by RN_i.

(b). *Pseudo valid output*: The recognition network may generate an output which follows the coding rule (1) or (2), but does not satisfy a given error tolerance. We call such an output the pseudo valid output. In the learning phase, we can easily judge whether an actual output is a pseudo valid output or not according to the following rule:

$$\exists j_{\,j \neq h}(x^{\mathrm{RN}_i}_{kj} \geq x^{\mathcal{O}}_{\text{high}}) \wedge \forall l_{\,l \neq j}(x^{\mathrm{RN}_i}_{kl} \leq x^{\mathcal{O}}_{\text{low}}) \vee \forall l(x^{\mathrm{RN}_i}_{kl} \leq x^{\mathcal{O}}_{\text{low}}) \quad \text{for } j \text{ and } l \in B_i, \tag{4}$$

where the desired output of the hth unit satisfies $\bar{x}^{\mathcal{O}}_{kh} \geq x^{\mathcal{O}}_{\text{high}}$. But after the learning, we can not use the above rule to judge whether an actual output is a pseudo valid output or not because there is no desired output that can be used. Therefore, to differentiate valid outputs from pseudo valid outputs must be achieved by learning.

(c). *Invalid output*: Otherwise.

For example, if the desired output pattern is (0, 0, 1), $\delta = 0.2$, $x^{\mathcal{O}}_{\text{high}} = 0.8$, and $x^{\mathcal{O}}_{\text{low}} = 0.2$, 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 an invalid output.

2.4 Decomposition of Control Task

The purpose of control network (CN_i) in the ith RC-form sieving module is used to learn the task ($\boldsymbol{C}_i$) that is to differentiate the valid outputs from pseudo valid outputs generated by RN_i. In PMSN, we assume that CN_i always learns $\boldsymbol{C}_i$ successfully in the learning process. Even though $\boldsymbol{C}_i$ is a two-class problem, it may be a very difficult classification task. From our experience, the training set $\boldsymbol{C}_i$ is imbalanced, that is, the number of the valid outputs ($N_{\mathrm{vo},i}$) is much greater than the number of the pseudo valid outputs ($N_{\mathrm{pvo},i}$). It has been observed that the standard backpropagation algorithm converges very slowly for imbalanced two-class classification problems. A modified backpropagation algorithm has been proposed for training networks on imbalanced training sets [1]. Although it has been shown that the modified algorithm converges much faster than the standard one, large and complex imbalanced classification problems still remain intractable.

In this paper, we present a multiple control network architecture for dealing with this problem. The basic idea behind this architecture is that $\boldsymbol{C}_i$ is decomposed into a finite set of manageable subtasks $\boldsymbol{c}_{i1}, \boldsymbol{c}_{i2}, \cdots, \boldsymbol{c}_{i\tau_i}$ ($\tau_i > 1$), where τ_i is determined by the ratio of $N_{\mathrm{vo},i}$ to $N_{\mathrm{pvo},i}$. Two main objectives of the decomposition are to lower the imbalanced ratio and to reduce the size and complexity of the training set. Various strategies may be used to implement the decomposition. The decomposition strategy may affect learning speed and generalization performance. In this paper we use random strategy to decompose $\boldsymbol{C}_i$. Let $\boldsymbol{u}_{ij}$ be the training input set corresponding to $\boldsymbol{c}_{ij}$. $\boldsymbol{u}_{ij}$ is defined as follows.

$$\boldsymbol{u}_{ij} = \boldsymbol{p}_i + \boldsymbol{v}_{ij} \quad \text{for } j = 1, 2, \cdots, \tau_i, \tag{5}$$

where $\boldsymbol{v}_{ij}$ is randomly selected from $\boldsymbol{V}_i$ and $\bar{\bar{\boldsymbol{v}}}_{ij} \approx N_{\mathrm{vo},i}/\tau_i$, $\boldsymbol{V}_i$ ($\boldsymbol{V}_i = \cup_{j=1}^{\tau_i} \boldsymbol{v}_{ij}$) and $\boldsymbol{p}_i$ are the training input sets corresponding to the valid and the pseudo valid outputs produced by RN_i, respectively, $\boldsymbol{v}_{ij} \subset \boldsymbol{V}_i$, $\boldsymbol{v}_{ij} \cap \boldsymbol{v}_{ik} = \emptyset$ for $j \neq k$, $\bar{\bar{\boldsymbol{V}}}_i = N_{\mathrm{vo},i}$, and $\bar{\bar{\boldsymbol{p}}}_i = N_{\mathrm{pvo},i}$.

2.5 Control Circuit

The outputs produced by a recognition network are classified and controlled by the control circuits as drawn by thin lines in Figs. 2(a) and 2(b). The output control circuit in the ith RC-form sieving module consists of an output judgment unit (OJU), multiple control networks (CN_{i1}, CN_{i2}, $\cdots$, $\mathrm{CN}_{i\tau_i}$), a minimum output selecting unit (MOU), an AND gate, two OR gates, and a logical switch.

(a) The output judgment unit is used to differentiate the invalid output from the valid and pseudo valid outputs. OJU in SM_i generates 1 or 0 according to

$$\mathcal{O}_{\mathrm{OJU}} = \begin{cases} 1, & \text{if } \boldsymbol{x}_k^{\mathrm{RN}_i} \text{ is a valid or pseudo} \\ & \text{valid output;} \\ 0, & \text{if } \boldsymbol{x}_k^{\mathrm{RN}_i} \text{ is an invalid output,} \end{cases} \tag{6}$$

where $\mathcal{O}_{\mathrm{OJU}}$ is the output of OJU_i. It should be noted that to distinguish the invalid outputs from the other two kinds of outputs is performed independently of the learning task, because the invalid outputs can be judged according to a given output representation scheme.

(b) The control network is used to differentiate the valid outputs from pseudo valid outputs. Since after learning there is no any desired output that can be used to judge whether an actual output is a valid output or a pseudo valid output, the control network must be trained. Its training output is set to 1 or 0 as follows.

$$\mathcal{O}_{\mathrm{CN}_{ij}} = \begin{cases} 0, & \text{if } \boldsymbol{x}_k^{\mathrm{RN}_i} \text{ is a valid output;} \\ 1, & \text{if } \boldsymbol{x}_k^{\mathrm{RN}_i} \text{ is a pseudo valid} \\ & \text{output,} \end{cases} \tag{7}$$

where $\mathcal{O}_{\mathrm{CN}_{ij}}$ is the output of the jth control subnetwork in SM_i.

(c) The minimum output selection unit MOU is used to choose minimum value from the outputs of trained multiple control networks, that is,

$$\mathcal{O}_{\mathrm{MOU}} = \begin{cases} 1, & \text{if } \mathrm{Min}\{\mathcal{O}_{\mathrm{CN}_{i1}}, \cdots, \mathcal{O}_{\mathrm{CN}_{i\tau_i}}\} \\ & \text{is greater than } \theta \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $\mathcal{O}_{\mathrm{MOU}}$ is the output of MOU, and θ is a threshold constant.

(d) The logical switch works as follows: If its control input is "1", then the data is blocked by it. Otherwise, the data passes through it.

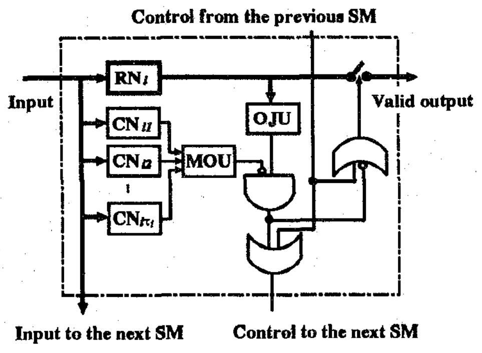
To achieve parallel processing, the priority to each SM is introduced in PMSN-MC. The priority is implemented by means of the output control circuits. In PMSN-MC, SM_i has higher priority than SM_j for $j > i$.

3 Learning Algorithm

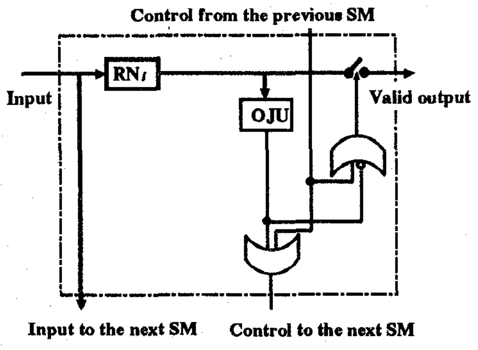
Let T_1 be a set of t_1 training samples:

$$T_1 = \{(\bar{\boldsymbol{x}}_k^{\mathcal{I}}, \bar{\boldsymbol{x}}_k^{\mathcal{O}}) \mid \text{for } k = 1, 2, \cdots, t_1\} \tag{9}$$

where $\bar{x}_k^{\mathcal{I}} \in \mathbf{R}^{N_{\mathcal{I}}}$ and $\bar{x}_k^{\mathcal{O}} \in \mathbf{R}^{N_{\mathcal{O}}}$ are the input and the desired output of the kth sample, respectively. Suppose that the number of iterations for training RN_i is bounded at most by K. The multi-sieving learning algorithm for training PMSN-MCs can be described as follows:



(a)



(b)

Figure 2: Two forms of sieving modules in PMSN-MC: the RC-form (a) and the R-form (b). In this figure and in Figure 4, crossing lines do not represent connections unless there is a dot on the intersection.

Step 1 : Initially, one recognition network, namely RN_1, is trained on the original set T_1 up to K iterations. 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. (3) and (4), respectively.

Step 3 : If $\sum_{i=1}^{m} N_{\mathrm{vo},i} = t_1$, i.e., if all t_1 samples are learned by SM_1 ,SM_2, $\cdots$, SM_m, then the training is completed.

Step 4 : If $N_{\mathrm{pvo},m} = 0$, i.e., if there is no pseudo valid output, then the control network is unnecessary. Go to Step 6.

Step 5 : If $N_{\mathrm{pvo},m} > 0$, i.e., if there exist $N_{\mathrm{pvo},m}$ pseudo valid outputs, then multiple control networks CN_{m1}, CN_{m2}, $\cdots$, CN_{mr_i} are selected. CN_{ij} is trained on the set c_{ij} until all of the samples are classified correctly.

$$c_{ij} = \{(\bar{x}_k^{\mathcal{I}},\ \bar{x}_k^{\mathcal{O}}) \mid \text{for } \bar{x}_k^{\mathcal{I}} \in u_{ij},$$

where $\bar{x}_k^{\mathcal{I}} \in \mathbf{R}^{N_{\mathcal{I}}}$ is the input whose output is a valid or a pseudo valid output, and $\bar{x}_k^{\mathcal{O}} \in \mathbf{R}^1$ is the desired output which is determined by

$$\bar{x}_k^{\mathcal{O}} = \begin{cases} 0, & \text{if the actual output of } \bar{x}_k^{\mathcal{I}} \text{ is} \\ & \text{a valid output;} \\ 1, & \text{if the actual output of } \bar{x}_k^{\mathcal{I}} \text{ is a} \\ & \text{pseudo valid output} \end{cases}$$

Step 6 : Freeze all of the parameters of RN_m and CN_m (if it exists), remove $N_{\mathrm{vo},m}$ samples which have been successfully classified by RN_m from T_m, and create a new training set consisting of t_{m+1} ($t_{m+1} = t_m - N_{\mathrm{vo},m}$) samples T_{m+1} ($T_{m+1} \subset T_m$), which are misclassified by RN_m.

Step 7 : If CN_m exists, construct SM_m in the RC-form. Otherwise construct SM_m in R-form.

Step 8 : Join SM_m to SM_{m-1} for $m > 1$ in the parallel structure as shown in Fig. 4.

Step 9 : Select RN_{m+1} and train it on T_{m+1} up to K iterations. Let $m = m + 1$ and go back to Step 2.

4 Simulation Results

In this section, we demonstrate the utility of PMSN-MC. For visualizing the input-output mappings formed by each network and the whole architecture, The "two-spirals" problem is treated [4]. In order to compare the performance of PMSN-MC with that of PMSN, the two-spirals problem is learned by both PMSN-MC and PMSN. In the following simulations, all the recognition and control networks are chosen to be the three-layer quadratic perceptron [5]. The standard backpropagation algorithm is used to train these networks [10].

The training inputs of the "two-spirals" problem consists of 194 points as shown in Fig. 3(a). Each recognition network has two input, five hidden and one output units. The training of RN_i is stopped after 10000 iterations if the sum of squared error (SSE) between the desired and the actual outputs cannot be less than the given value, 0.01.

Firstly, RN_1 is trained on the 194 data. After 10000 iterations, the training is stopped since SSE = 15.77. Presenting 194 training inputs to RN_1 again, we obtain $N_{vo,i} = 64$ and $N_{pvo,i} = 0$ according to Eqs. (3) and (4), that is, there are 64 training data (see Fig. 3(b)) have been successfully learned by RN_1 and there exists no any pseudo valid output. Consequently, SM_1 is selected as the R-form.

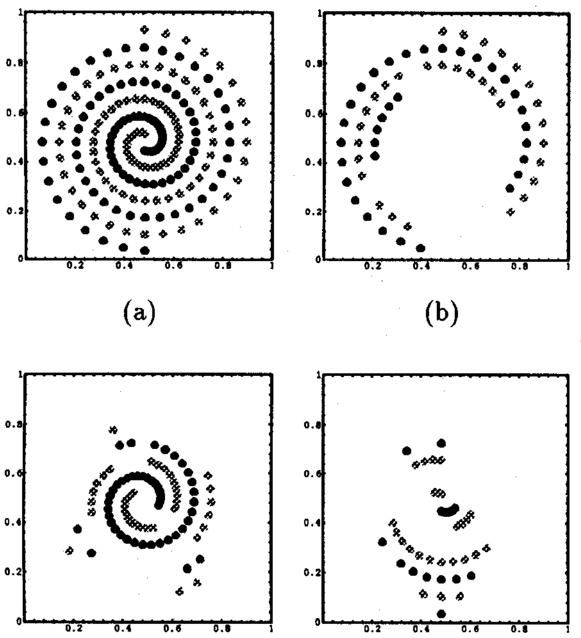

(a) (b) (c) (d)

Figure 3: The input patterns of the "two-spirals" problem (a), the training inputs learned by the first sieving module (b), the second sieving module (c), and the third sieving module (d). For black and grey points, the RN_i is required to generate output 0 and 1, respectively.

Secondly, RN_2 is trained on the remaining 130 data. After 10000 iterations, the training is stopped since SSE = 7.69. Presenting 130 data to RN_2 again, we obtain $N_{vo,2} = 87$ and $N_{pvo,2} = 8$, i.e., there are 87 training data have been correctly learned by RN_2 (see Fig. 3(c)) and there exist 8 pseudo valid outputs produced by RN_2. In PMSN, a single control network (CN_2) with ten hidden units is trained to differentiate 8 pseudo valid outputs from 87 valid output. The training input patterns for CN_2 is shown in Fig. 5(a). In PMSN-MC, the training inputs related to 87 valid outputs are randomly partitioned into three subsets, $\boldsymbol{v}_{21}$, $\boldsymbol{v}_{22}$ and $\boldsymbol{v}_{23}$. The number of elements in each of the subsets is 29. The training input patterns for three control sub-networks (CN_{21}, CN_{22} and CN_{23}) are shown in Figs. 5(b) through 5(d), respectively. The CPU time for training CN_2 and the total CPU time for training CN_{21}, CN_{22} and CN_{23} are about 21326 and 253 seconds, respectively, on a Sparc-20 workstation.

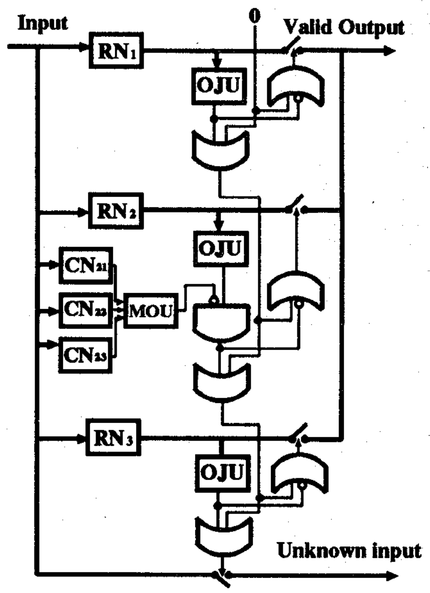


Figure 4: The PMSN-MC for learning the "two-spirals" problem, where the control signal to the first sieving modules is set to "0".

Finally, RN_3 is trained on the remaining 43 data (see Fig. 3(d)). After 8009 iterations, the training is stopped since SSE is less than 0.01. Presenting the 43 data to RN_3 again, we obtain $N_{vo,3} = 43$ and $N_{pvo,3} = 0$. Since $N_{vo,1} + N_{vo,2} + N_{vo,3} = 194$, the training is finished.

After the training, the two-spirals problem is automatically decomposed into three subtasks and learned by a PMSN-MC with three sieving modules as shown in Fig. 4. The response plots of PMSN and PMSN-MC are illustrated in Figs. 6(a) and 6(b), respectively. From these two figures, we can see that the generalization performance of PMSN-MC and PMSN are very similar.

5 Conclusion

We have presented a parallel and modular multi-sieving neural network architecture with multiple control networks. By using this architecture, the learning of imbalanced two-class classification problem for control networks, a bottleneck problem in multi-sieving learning, can be dealt with efficiently. Consequently, automatic decomposition of large and complex learning tasks can be easily implemented using multi-sieving learning method.

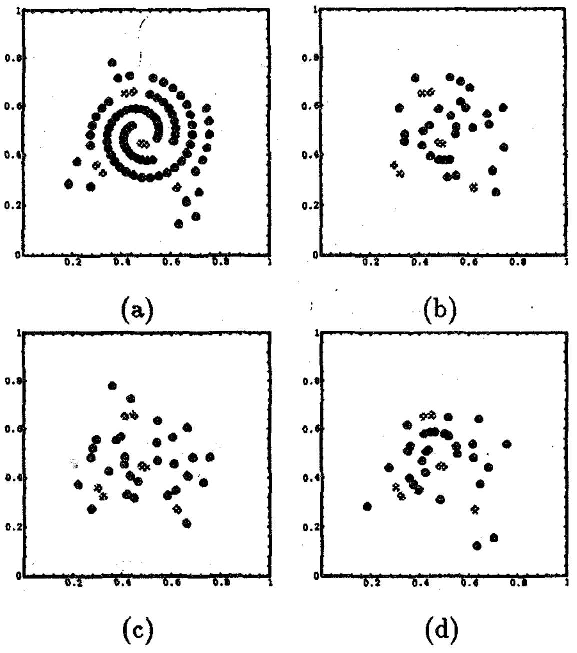


Figure 5: The training inputs for single control network, (a) and the training inputs for three control sub-networks, CN_{21} (b), CN_{22} (c) and CN_{23} (d), respectively. The black and grey points correspond to the inputs related to the valid and pseudo valid outputs produced by RN_2, respectively. For black and grey points, each control network is required to produce output 0 and 1, respectively.

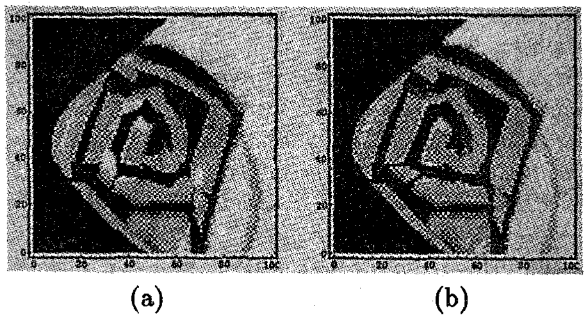


Figure 6: The response plots of PMSN (a) and PMSN-MC (b), respectively. Black and white represent output of "0" and "1", respectively, and gray represents intermediate value.

References

[1] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka, "An improved algorithm for neural network classification of imbalanced training sets", *IEEE Trans. on Neural Networks*, 1993, pp. 962-969.

[2] S. Fahlman and C. Lebiere, "The cascade-correlation learning architecture", *Tech. Rep. CMU-CS-90-100*, Carnegie-Mellon University, 1991.

[3] R. A. Jacobs, M. I. Jordan, M. I. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts", *Neural Computation*, vol. 3, 1991, pp. 79-87.

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

[5] B. L. Lu, Y. Bai, H. Kita, and Y. Nishikawa, "An efficient multilayer quadratic perceptron for pattern classification and function approximation", *Proc. of International Joint Conference on Neural Networks*, 1993, vol. 2, pp. 1385-1388, Nagoya.

[6] B. L. Lu, "Architectures, learning and inversion algorithms for multilayer neural networks", *PhD thesis*, Dept. of Electrical Engineering, Kyoto University, 1994.

[7] B. L. Lu, H. Kita, and Y. Nishikawa, "A multi-sieving neural network architecture that decomposes learning tasks automatically", *Proceedings of IEEE Conference on Neural Networks*, 1994, pp. 1319-1324, Orlando, FL.

[8] B. L. Lu, K. Ito, H. Kita, and Y. Nishikawa, "A parallel and modular multi-sieving neural network architecture for constructive learning", *Proceedings of IEE International Conference on Artificial Neural Networks*, 1995, pp. 92-97, Cambridge, UK.

[9] M. Mezard and J. P. Nada, "Learning in feedforward layered networks: the titling algorithm", *Journal of Physics A*, vol. 22, 1989, pp. 2191-2203.

[10] D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing*, vol. 1, Cambridge, MA: MIT Press, 1986.