Task Decomposition Based on Class Relations: A Modular Neural Network Architecture for Pattern Classification

Bao-Liang Lu and Masami Ito

Bio-Mimetic Control Research Center,
the Institute of Physical and Chemical Research (RIKEN)
3-8-31 Rokuban, Atsuta-ku, Nagoya 456, Japan
lbl@nagoya.bmc.riken.go.jp; itom@nagoya.bmc.riken.go.jp

Abstract. In this paper, we propose a new methodology for decomposing pattern classification problems based on the class relations among training data. We also propose two combination principles for integrating individual modules to solve the original problem. By using the decomposition methodology, we can divide a K-class classification problem into $\binom{K}{2}$ relatively smaller two-class classification problems. If the two-class problems are still hard to be learned, we can further break down them into a set of smaller and simpler two-class problems. Each of the two-class problem can be learned by a modular network independently. After learning, we can easily integrate all of the modules according to the combination principles to get the solution of the original problem. Consequently, a K-class classification problem can be solved effortlessly by learning a set of smaller and simpler two-class classification problems in parallel.

1 Introduction

One of the most important difficulties in using artificial neural networks for solving large-scale, real-world problems is how to divide a problem into smaller and simpler subproblems; how to assign a modular network to learn each of the subproblems independently; and how to combine the individual modules to get the solution of the original problem. In the last several years, many researchers have studied modular neural network systems for dealing with this problem, for example see [8, 3, 2, 1, 7]. Up to now, various problem decomposition methods have been developed based on the divide-and-conquer strategy. These methods can be roughly classified into three classes as follows.

Explicit decomposition: Before learning, a problem is divided into a set of subproblems by a designer who should have some domain knowledge and deep prior knowledge concerning the decomposition of the problem. Several modular systems have been developed based on this decomposition method, see for instance [10, 4]. The limitation of this method is that sufficient prior knowledge concerning the problem is necessary.

Class decomposition: Before learning, a problem is broken down into a set of subproblems according to the inherent relations among training data. Anand

et al. [1] first introduced this method for decomposing a K-class classification problem into K two-class problems by using the class relations among the training data. In contrast to the explicit decomposition, this method only needs some common knowledge concerning the training data,

Automatic decomposition: A problem is decomposed into a set of subproblems with the progressing of the learning. Most of the existing decomposition methods fall into this category, see for instance [2, 7]. From computational complexity's point of view, the former two methods are more efficient than this one because the problems have been decomposed into subproblems before learning, and therefore, they are suitable for solving large-scale and complex problems. The advantage of this method is that it is more general than the former ones because it can work when prior knowledge concerning the problem is absent.

In this paper, we propose a new methodology for decomposing classification problems. The basic idea behind this methodology is to use the class relations among the training data, similar to the method developed by Anand *et al.* [1]. In comparison with Anand's method, our methodology has two main advantages as follows. (a) The two-class problem obtained by our method is to discriminate between every pair classes, i.e., class $\mathcal{C}_i$ and class $\mathcal{C}_j$ for $i = 1, \cdots, K$ and $j = i+1$. The existence of the training data of the other $K-2$ classes is ignored. Therefore, the number of training data for each of the two-class problems is $2N$. However, the two-class problem obtained by Anand's method has to discriminate between one class and the remaining classes. Therefore, the number of training data for each of the two-class problems is $K \cdot N$. When K is large, learning of the two-class problems obtained by Anand's method may be still problematic. Here, for simplicity of description, the assumption we made is that each of the classes has the same number of training data N. (b) By using our method, the two-class problem can be further divided into $N_i \cdot N_j$ smaller and simpler two-class problems, where N_i and N_j are the numbers of training subsets belonging to $\mathcal{C}_i$ and $\mathcal{C}_j$, respectively. However, Anand's method can not be applied to decomposing two-class problems. Since the two-class problems obtained by our method can be much smaller and simpler than those obtained by Anand's method, it is easier to assign a smaller modular network to learn each of the two-class problems. We also propose two combination principles for integrating individual modules to solve the original problem. After training each of the two-class problem with a modular network, we can easily integrate all of the modules according to the combination principles to create a solution to the original problem. Consequently, a K-class classification problem can be solved effortlessly by learning a set of smaller and simpler two-class problems in parallel.

The remainder of the article is organized as follows. In Section 2, we present a new decomposition methodology. In Section 3, we introduce three integrating units for constructing modular networks and describe two combination principles. Section 4 gives several examples and simulation results. Finally, conclusions are given in Section 5.

2 The Task Decomposition Methodology

The decomposition of a task is the first step to implement a modular neural network system. In this section, we present a new methodology for decomposing a K-class classification problem into a set of smaller and simpler two-class classification problems.

2.1 Decomposition of K-class problems

We address K-class ($K > 1$) classification problems. Suppose that grandmother cells are used as output representation. Let $\mathcal{T}$ be the training set for a K-class classification problem:

$$\mathcal{T} = \{(X_l, Y_l)\}_{l=1}^{L}, \tag{1}$$

where $X_l \in \mathbf{R}^d$ is the input vector, and $Y_l \in \mathbf{R}^K$ is the desired output.

A K-class problem can be divided into K two-class problems [1]. The training set for each of the two-class problems is defined as follows:

$$\mathcal{T}_i = \{(X_l, y_l^{(i)})\}_{l=1}^{L} \quad \text{for } i = 1, \cdots, K \tag{2}$$

where $X_l \in \mathbf{R}^d$ and $y_l^{(i)} \in \mathbf{R}^1$. The desired output $y_l^{(i)}$ is defined as:

$$y_l^{(i)} = \begin{cases} 1 - \epsilon & \text{if } X_l \text{ belongs to class } \mathcal{C}_i \\ \epsilon & \text{if } X_l \text{ belongs to } \bar{\mathcal{C}}_i \end{cases} \tag{3}$$

where ϵ is a small positive real number, $\bar{\mathcal{C}}_i$ denotes all the classes except $\mathcal{C}_i$. That is, $\bar{\mathcal{C}}_i$ is $\mathcal{C}_i$'s complement.

If the original K-class problem is large and complex, learning of the two-class problems as defined in Eq. (2) may be still problematic. One may ask: whether can the two-class classification problems be further decomposed into simpler two-class problems ? We will give an answer to this question in the remainder of the article.

2.2 Decomposition of two-class problems

From Eq. (1), the input vectors can be easily partitioned into K sets:

$$\mathcal{X}_i = \{X_l^{(i)}\}_{l=1}^{L_i} \quad \text{for } i = 1, 2, \cdots, K, \tag{4}$$

where $X_l^{(i)} \in \mathbf{R}^d$ is the input vector, all of the $X_l^{(i)} \in \mathcal{X}_i$ have the same desired outputs, and $\sum_{i=1}^{K} L_i = L$. Note that this partition is unique.

We suggest that the two-class problems as defined in Eq. (2) can be further divided into $K - 1$ smaller two-class problems. The training set for each of the smaller two-class problems is defined as follows:

$$\mathcal{T}_{ij} = \{(X_l^{(i)}, 1 - \epsilon)\}_{l=1}^{L_i} \cup \{(X_l^{(j)}, \epsilon)\}_{l=1}^{L_j} \quad \text{for } j = 1, \cdots, K \text{ and } j \neq i \tag{5}$$

where $X_l^{(i)} \in \mathcal{X}_i$ and $X_l^{(j)} \in \mathcal{X}_j$ are the input vectors belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively. For task $\mathcal{T}_{ij}$, the existence of the training data belonging to the other $K-2$ classes is ignored.

From Eq. (5), we see that partitioning of the two-class problem as defined in Eq. (2) into $K-1$ smaller two-class problem is simple and straightforward. No domain specialists or prior knowledge concerning the decomposition of the learning problems are required. Consequently, any designer can perform this decomposition easily if he or she knows the number of training patterns belonging to each of the classes.

From Eq. (5), we see that a K-class problem can be broken down into $K \cdot (K-1)$ two-class problems, which are represented as a $K \times K$-matrix as follows:

$$\begin{bmatrix} \emptyset & \mathcal{T}_{12} & \mathcal{T}_{13} & \dots & \mathcal{T}_{iK} \\ \mathcal{T}_{21} & \emptyset & \mathcal{T}_{23} & \dots & \mathcal{T}_{2K} \\ \dots & \dots & \dots & \dots & \dots \\ \mathcal{T}_{K1} & \mathcal{T}_{K2} & \mathcal{T}_{K3} & \dots & \emptyset \end{bmatrix} \tag{6}$$

where $\emptyset$ represents empty set.

In fact, among the the above problems, only $\binom{K}{2}$ two-class problems in the upper triangular are different, and other $\binom{K}{2}$ ones in the lower triangular can be solved by inverting the former ones by using the INV units (see Section 3). Therefore, the number of two-class problems that need to be learned can be reduced to $\binom{K}{2}$. Comparing Eq. (5) with Eq. (2), we see that the two-class problem defined in Eq. (5) is much smaller than that defined in Eq. (2) if the K is large and the number of patterns for each of the K-classes is roughly equal.

2.3 Fine decomposition of two-class problems

Even though a K-class problem can be broken down into $\binom{K}{2}$ relatively smaller two-class problems, some of them may be still hard to be learned: for instance, the "two-spirals" problem [5]. In order to deal with this problem, we propose a method for further decomposing the two-class problem $\mathcal{T}_{ij}$ as defined in Eq. (5) into a set of smaller and simpler two-class problems.

Assume that the input set $\mathcal{X}_i$ is further partitioned into N_i ($N_i \geq 1$) subsets:

$$\mathcal{X}_{ij} = \{X_l^{(ij)}\}_{l=1}^{L_i^{(j)}} \quad \text{for } j = 1, \cdots, N_i, \tag{7}$$

where $X_l^{(ij)} \in \mathbf{R}^d$ is the input vector and $\sum_{j=1}^{N_j} L_i^{(j)} = L_i$. This partition is not unique in general. One can give a partition randomly or by using prior knowledge concerning the decomposition of the learning problems.

The training set for each of the smaller and simpler two-class problems is defined as follows:

$$\begin{aligned} \mathcal{T}_{ij}^{(uv)} = &\{(X_l^{(iu)},\ 1-\epsilon)\}_{l=1}^{L_i^{(u)}} \cup \{(X_l^{(jv)},\ \epsilon)\}_{l=1}^{L_j^{(v)}} \\ &\text{for } u = 1, \cdots, N_i,\ v = 1, \cdots, N_j, \text{ and } j \neq i \end{aligned} \tag{8}$$

where $X_l^{(iu)} \in \mathcal{X}_{iu}$ and $X_l^{(jv)} \in \mathcal{X}_{jv}$ are the input vectors belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively.

3 The Modular Network Architecture

After solving each of the smaller two-class problems as defined in Eq. (5) or Eq. (8) by a modular network, we need to organize the individual modules and construct a modular system to get the solution of the original problem. In this section, we will first introduce three integrating units for constructing the modular networks, and then we will give two combination principles for integrating the individual modules.

3.1 Three Integrating Units

Before describing our modular neural network architecture, we introduce three integrating units, namely MIN, MAX, and INV respectively.

The basic function of a MIN unit is to find a minimum value from its multiple inputs. The transfer function of a MIN unit is given by

$$q = \text{Minimize}\{p_1, \cdots, p_n\} \tag{9}$$

where $p_1, \cdots, p_n$ and q are the inputs and output, respectively, $p_i \in \mathbf{R}^1$ for $i = 1, \cdots, n$, and $q \in \mathbf{R}^1$.

The basic function of a MAX unit is to find a maximum value from its multiple inputs. The transfer function of a MAX unit is given by

$$q = \text{Maximize}\{p_1, \cdots, p_n\} \tag{10}$$

where $p_1, \cdots, p_n$ and q are the inputs and output, respectively.

The basic function of an INV unit is to invert its single input. The transfer function of an INV unit is given by

$$q = b - p \tag{11}$$

where b, p and q are the upper limit of its input, input, and output, respectively.

3.2 The Combination Principles

Suppose that each of the two-class problems has been learned by a modular network completely. One may ask a question: how to combine the outputs of the individual modules to get the solution of the whole problem ? In this subsection, we will present two combination principles which give the designer a systematic method for organizing the modules.

Minimization Principle: *The modules, which were trained on the same training inputs corresponding to the desired outputs* $1-\epsilon$*, should be integrated by the MIN unit.*

Consider the two-class problems $\mathcal{T}_{i1}, \mathcal{T}_{i2}, \cdots, \mathcal{T}_{iK}$ as defined in Eq. (5). These problems have the same training inputs corresponding to the desired outputs $1-\epsilon$. Suppose that the $K-1$ modules, which are represented as $\mathcal{M}_{i1}$, $\mathcal{M}_{i2}$, $\cdots$, $\mathcal{M}_{iK}$, were trained, respectively, on $\mathcal{T}_{i1}$, $\mathcal{T}_{i2}$, $\cdots$, $\mathcal{T}_{iK}$. According to the minimization principle, we can organize the $K \cdot (K-1)$ modules into a modular network as illustrated in Fig. 1(a), where, for simplicity of illustration, the assumption we made is that all of the $K \cdot (K-1)$ two-class problems as defined in Eq. (5) are learned and no INV unit is used.

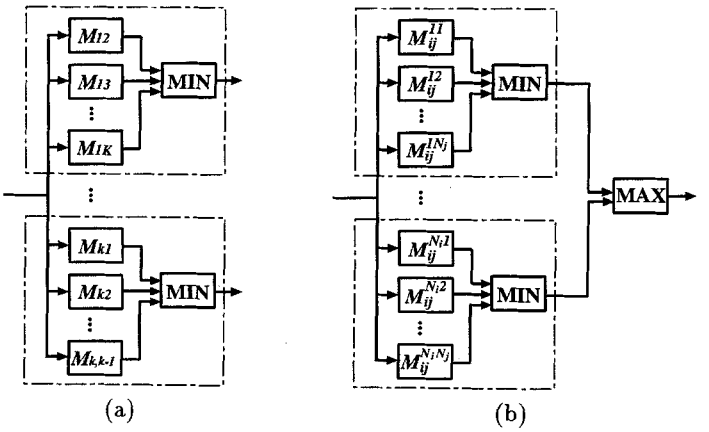

Fig. 1. The organization of the $K \cdot (K-1)$ modules by using the MIN units (a) and the organization of the $N_i \cdot N_j$ modules by using the MIN and MAX units (b).

Maximization Principle: *The modules, which were trained on the same training inputs corresponding to the desired outputs* ϵ*, should be integrated by the MAX unit.*

Consider the combination of the modules which were trained on the following $N_i \cdot N_j$ two-class problems as defined in Eq. (8).

$$\begin{matrix} \mathcal{T}_{ij}^{(11)} & \mathcal{T}_{ij}^{(12)} & \cdots \mathcal{T}_{ij}^{(1,N_j)} \\ \mathcal{T}_{ij}^{(21)} & \mathcal{T}_{ij}^{(22)} & \cdots \mathcal{T}_{ij}^{(2,N_j)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & \\ \mathcal{T}_{ij}^{(N_i,1)} & \mathcal{T}_{ij}^{(N_i,2)} & \cdots \mathcal{T}_{ij}^{(N_i,N_j)} \end{matrix} \tag{12}$$

According to the decomposition method defined in Eq. (8), the N_j training sets in each of row of Eq. (12) have the same training inputs corresponding to the desired outputs $1-\epsilon$. In contrast, the N_i training sets in each column of Eq. (12)

have the same training inputs corresponding to the desired outputs ϵ. Following the minimization and maximization principles, the $N_i \cdot N_j$ modules that were trained on the $N_i \cdot N_j$ two-class problems can be organized as illustrated in Fig. 1(b).

4 Examples and Simulations

To evaluate the effectiveness of the proposed decomposition methodology, the two combination principles, and the modular network architecture, several benchmark learning problems have been simulated in this section. In the following simulations, the structure of all the nonmodular and modular networks are chosen to be the three-layer quadratic perceptrons with one hidden layer [6]. All of the networks are trained by the back-propagation algorithm [9]. The momentums are set all to 0.9. The learning rates are selected through practical experiments. They are optimal for fast convergence. For each of the nonmodular and modular networks, training was stopped when the mean square error for each network was reduced to 0.01. A summary of the simulation results is shown in Table 1, where "Max." means the maximum CPU time required to train any modular network. All of the simulations were performed on a SUN Sparc-20 workstation.

Two-Spirals Problem: The "two-spirals" problem [5] is chosen as a benchmark for this study because it is an extremely hard two-class problem for the conventional backpropagation networks and the mapping from input to output formed by each of the modules is visible.

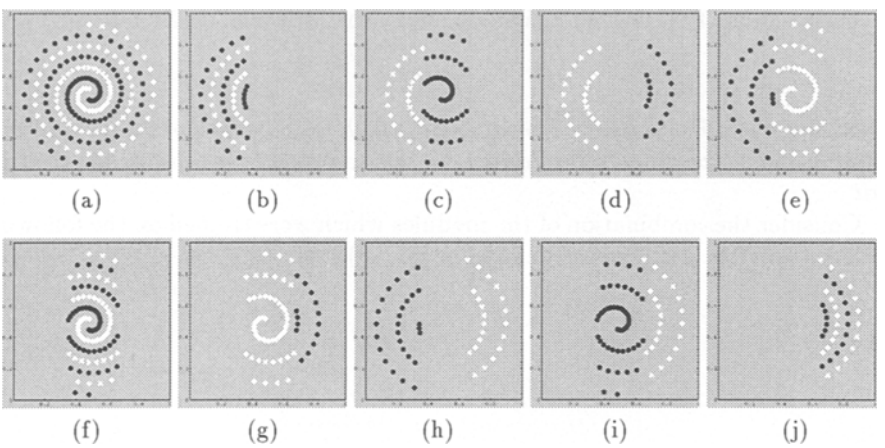

Fig. 2. The training inputs for the original two-spirals problem (a). The training inputs for the nine subproblems (b) through (j), respectively. The black and white points represent the desired outputs of "0" and "1", respectively.

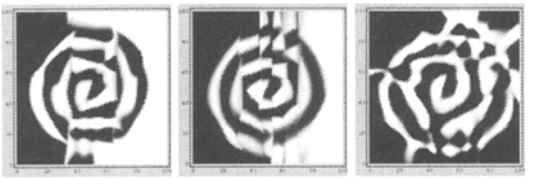

Fig. 3. The responses of the modular network with the 9 modules (a), the modular network with the 36 modules (b), and the single network with 40 hidden units (c). Black and white represent the outputs of "0" and "1", respectively.

The 194 training inputs for the original two-spirals problem are shown in Fig. 2(a). We performed three comparative simulations on this problem. In the first simulation, the original problem was divided into nine subproblems by partitioning the input variable through the axis of abscissas into three overlapping intervals. The training inputs for the nine subproblems are shown in Figs. 2(b) through 2(j), respectively. All of the nine modular networks were selected to be five hidden units except that the fifth module was selected to be twenty-five hidden units because the fifth task (see Fig. 2(f)) is the hardest problem to be learned in the nine problems. The combination of the outputs of the nine trained modules is shown in Fig. 3(a). In the second simulation, the original problem was divided into 36 subproblems by partitioning the input variable through the axis of abscissas into 6 overlapping intervals. The numbers of hidden units of the 1st, the 8th, the 15th, the 22nd and the 29th modules were chosen to be 10, and the others were chosen to be 1. The response of the modular network which consists of 36 trained modules is shown in Fig. 3(b). For comparing with the above results, this problem was also learned by a single network with 40 hidden units. After 200,000 iterations, the mean square error was still about 0.57. The response of the single network is shown in Fig. 3(c). All of the CPU times required to train the single and modular networks are shown in Table 1.

Table 1. Performance comparison of nonmodular and the proposed modular networks

Task	Network	♯ Modules	CPU time		Success rate (%)	
			Max.	Total	Training data	Test data
Two-spirals	Nonmodular	1	105447	105447	99.48%	-
	Modular	9	5513	5983	100.00%	-
	Modular	36	648	1439	100.00%	-
Image	Nonmodular	1	50828	50828	99.95%	91.19%
	Modular	21	350	1121	100.00%	90.76%
Vehicle	Nonmodular	1	134971	134971	99.76%	72.34%
	Modular	6	3456	4567	100.00%	73.05%

Image Segmentation: The image segmentation problem was obtained from the University of California at Irvine (UCI) repository of machine learning databases. This real problem consists of 210 training data and 2100 test data. The number of attributes is 19 and the number of classes is 7. The original problem is decomposed into $\binom{7}{2}$ two-class problems according to the decomposition

method defined in Eq. (5). Each of the two-class problems consists of 60 training data. Each of the 21 two-class problems was learned by a modular network with 3 hidden units. The original problem was also learned by a single network with 10 hidden units. The simulation results are shown in Table 1.

Vehicle Classification: This real classification problem was also obtained from UCI repository of machine learning databases. The problem is to classify a given silhouette as one of four types of vehicle by using a set of features extracted from the silhouette. We divided the original data set into training and test sets. Each of the two sets consists of 423 data. The number of attributes is 18 and the number of classes is 4. The original problem was decomposed into $\binom{4}{2}$ two-class problems. All of the 6 modules were selected to be 4 hidden units, except that the module used to train on $\mathcal{T}_{23}$ was selected to be 8 hidden units. The 6 trained modules are organized as illustrated in Fig. 4. This original problem was also learned by a single network with 24 hidden units. The simulation results are shown in Table 1.

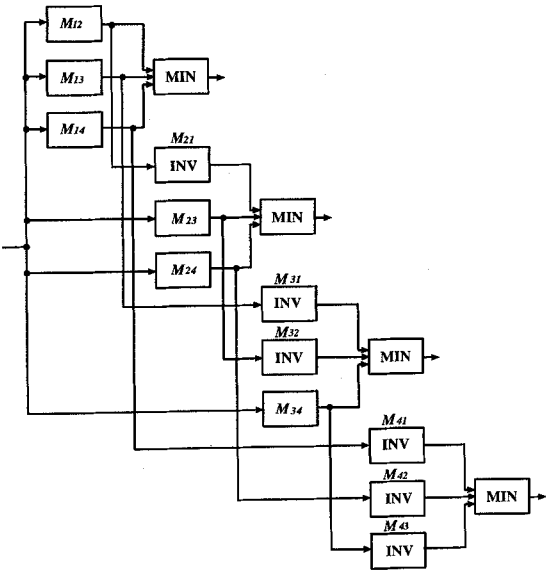

Fig. 4. The modular network architecture for learning the Vehicle classification problem. Corss lines do not represent connections unless there is a dot on the intersection.

5 Conclusions

In this paper, we have proposed a new decomposition methodology, two combination principles for integrating modules, and a new modular neural network architecture. The basic idea of the methodology is based on the class relations among

the training data. Given a K-class classification problem, by using the proposed decomposition methodology, we can divide the problem into a set of smaller and simpler two-class problems. Several attractive features of this methodology can be summarized as follows: (a) we can break down a problem into a set of smaller subproblems even though we are not domain specialists or we have no any prior knowledge concerning the decomposition of the problem; (b) training of each of the two-class problems can be greatly simplified and achieved independently; and (c) different network structures or different learning algorithms can be used to learn each of the problems. The two combination principles gives us a systematic method for organizing the individual modules. By using three integrating units, we can combine the outputs of all the individual modules to create a solution to the original problem. The simulation results (see Table 1) indicate that (a) the speedups of up to one order of magnitude can be obtained with our modular network architecture and (b) the generalization performance of trained single and modular networks are about the same. The importance of the proposed decomposition methodology lies in the fact that it provides us a promising approach to solving large-scale, real-world pattern classification problems.

References

1. Anand, R., Mehrotra, K. G., Mohan, C. K., and Ranka, S.: Efficient classification for multiclass problems using modular neural networks, *IEEE Transaction on Neural Networks*, 1995, **6**(1), 117-124.
2. Jacobs, R. A., Jordan, M. I., Nowlan, M. I., and Hinton, G. E.: Adaptive mixtures of local experts, *Neural Computation*, 1991, **3**, 79-87.
3. Hrycej, T.: *Modular Learning in Neural Networks*, 1992, John-Wiley & Sons, Inc.
4. Jenkins, R., and Yuhas, B.: A simplified neural network solution through problem decomposition: The case of the truck backer-upper, *IEEE Transaction on Neural Networks*,1993, **4**(4), 718-722.
5. Lang, K., and Witbrock, M.: Learning to tell two spirals apart, *Proceedings of 1988 Connectionist Models Summer School*, 1988, 52-59. Morgan Kaufmann.
6. Lu, B. L., Bai, Y., Kita, H., and Nishikawa, Y.: An efficient multilayer quadratic perceptron for pattern classification and function approximation, *Proceedings. of International Joint Conference on Neural Networks*, Nagoya, 1993, 1385-1388.
7. Lu, B.-L., Kita, H., and Nishikawa, Y.: A multi-sieving neural network architecture that decomposes learning tasks automatically, *Proceedings of IEEE Conference on Neural Networks*, 1994, 1319-1324.
8. Murre, J. M. J.:*Learning and Categorization in Modular Neural Networks*, 1992, Harvester Wheatsheaf.
9. Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1996, 1, D. E. Rumelhart, J. L. McClelland, and PDP Research Group eds, MIT Press.
10. Thiria, S., Mejia, C., Badran, F., and Crepon, M.: Multimodular architecture for remote sensing operations, *Advances in Neural Information processing Systems 4*, 1992, 675-682.