

Reduced Pattern Training Based on Task Decomposition Using Pattern Distributor

Sheng-Uei Guan, Chunyu Bao, and TseNgee Neo

Abstract—Task decomposition with pattern distributor (PD) is a new task decomposition method for multilayered feedforward neural networks (NNs). Pattern distributor network is proposed that implements this new task decomposition method. We propose a theoretical model to analyze the performance of pattern distributor network. A method named reduced pattern training (RPT) is also introduced, aiming to improve the performance of pattern distribution. Our analysis and the experimental results show that RPT improves the performance of pattern distributor network significantly. The distributor module's classification accuracy dominates the whole network's performance. Two combination methods, namely, crosstalk-based combination and genetic-algorithm (GA)-based combination, are presented to find suitable grouping for the distributor module. Experimental results show that this new method can reduce training time and improve network generalization accuracy when compared to a conventional method such as constructive backpropagation or a task decomposition method such as output parallelism (OP).

Index Terms—Crosstalk-based combination, full pattern training (FPT), genetic-algorithm-based combination, pattern distributor, reduced pattern training (RPT), task decomposition.

I. INTRODUCTION

MULTILAYERED feedforward neural networks (NNs) have been widely used in solving classification problems. However, they still exhibit some drawbacks when applied to large-scale real-world problems. One common drawback is that large networks tend to introduce high internal interference because of the strong coupling among the hidden-layer weights [1]. The influences from two or more output units could cause the hidden-layer weights to compromise to nonoptimal values due to the interference in their weight-updating directions during the weight-updating process [2]. Various task decomposition methods have been proposed to overcome this drawback [2]–[10], [18]–[21], [23]–[26]. Instead of using a single, large feedforward network (classic network), task decomposition methods divide a problem into a set of smaller and simpler subproblems based on “divide-and-conquer.” The results obtained from solving these subproblems are integrated together and the solution for the original problem is obtained.

Anand *et al.* proposed a method that splits a K -class problem into K two-class subproblems [3]. Each subnetwork is trained to

learn one subproblem only. Therefore, each subnetwork is used to discriminate one class of patterns from patterns belonging to the remaining classes, thereby resulting in K -modules in the overall structure. Another method divides the K -class problem into $\binom{K}{2}$ two-class subproblems [4]. A module is designated to learn each subproblem while training patterns belonging to the other $K - 2$ classes are ignored. The final overall solution is obtained by integrating all the trained modules into a min-max modular network. A powerful extension to the aforementioned class decomposition method, output parallelism (OP), is proposed by Guan [2], [5]–[8]. Using OP, a problem can be divided into several subproblems as chosen, each of which is composed of the whole input vector and a fraction of the output vector. Each module (for one subproblem) is responsible for producing a fraction of the output vector of the original problem. These modules can be grown and trained in parallel. Instead of decomposing the problem with a high-dimensional output space into several subproblems, each with a low-dimensional output space, Lu decomposes the problem into several smaller size subproblems [10]. Patterns are classified by a rough sieve module (nonmodular network) at the beginning and those patterns that are not classified successfully will be presented to another sieve module. This process continues until all the patterns are classified correctly. The sieve modules are added to the network adaptively with the progress of training.

Although these methods are efficient, there are still some drawbacks associated with them. First, the methods proposed in [3] and [4] split the problem into a set of two-class subproblems. If the original K -class problem is complex (K is large), a large number of modules will be needed to learn the subproblems, and thus, resulting in excessive computational cost. Second, although the dimension (number of output class) of each subproblem in [2] and [3] is smaller than the original problem, the size of each subproblem's training pattern set is still as large as the original problem. Therefore, each module will have long training time and ineffective learning especially when the original problem is large with many training patterns. Last, the method proposed in [10] only reduces the size of the problem but not the dimension of the problem. The internal interferences (that exists within each module due to the coupling of output units) are not reduced.

In this paper, we propose a new task decomposition method called task decomposition with pattern distributor (PD) to overcome the drawbacks mentioned previously. A special module called distributor module is introduced in order to improve the performance of the whole network. The distributor module and the other modules in the PD network are arranged in a hierarchical structure. The distributor module has a higher position

Manuscript received November 23, 2005; revised November 9, 2006 and February 16, 2007; accepted February 20, 2007.

S.-U. Guan is with the School of Engineering and Design, Brunel University, Uxbridge, Middlesex UB8 3PH, U.K. (e-mail: steven.guan@brunel.ac.uk).

C. Bao and T. Neo are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119260, Singapore (e-mail: chunyubao@gmail.com).

Digital Object Identifier 10.1109/TNN.2007.899711

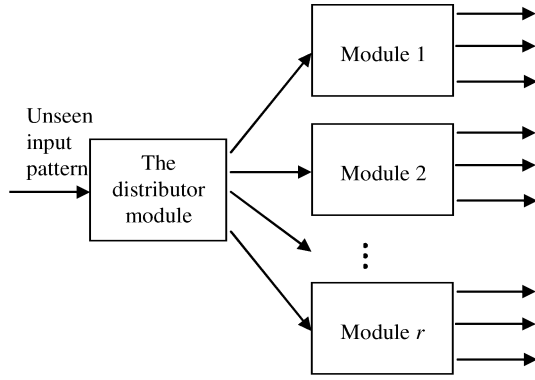


Fig. 1. Typical PD network.

as compared to the other modules in the network. This means an unseen input pattern will be recognized by the distributor module first. The structure of a typical PD network is shown in Fig. 1. Each output of the distributor module consists of a fraction of the overall output classes in the original problem. The PD method could shorten the training time and improve the generalization accuracy of a network compared with ordinary task decomposition methods.

In this paper, the PD method will be discussed in details. In Section II, a theoretical model is presented to compare the performance of a PD network with a typical task decomposition network—OP network. In Section III, we introduce the reduced pattern training (RPT) method to improve the PD network’s performance. Because of the importance of the distributor module, we present in Section IV two combination methods, crosstalk-and genetic-algorithm (GA)-based combinations, to find good grouping for the distributor module. In Section V, the experimental results are shown and analyzed. Conclusions are presented in Section VI.

II. THEORETICAL MODEL FOR THE PD NETWORK

There are two types of modules in a PD network, distributor and nondistributor modules (for simplicity, nondistributor modules are just called modules). Normally, a PD network consists of one distributor module and several nondistributor modules.

Class decomposition is often used to in solving classification problems. Compared with ordinary methods in which only an NN is constructed to solve the problem, class decomposition divides the problem into several subproblems and trains an NN module for each subproblem. Then, the results from these modules are integrated to obtain the solution for the original problem. OP is a typical class decomposition method. Here we present a model to show that the PD method has better performance than the OP method when the recognition rate of the distributor module is guaranteed.

Consider a classification problem with K output classes. To solve the problem, a PD network with one distributor module and r nondistributor modules is constructed. See Fig. 2 for details. There are r outputs in the distributor module and each nondistributor module is connected to an output of the distributor module. Each output of the distributor module consists of a combination of several classes. For an unknown pattern, the distributor module recognizes and dispatches it to only one of the

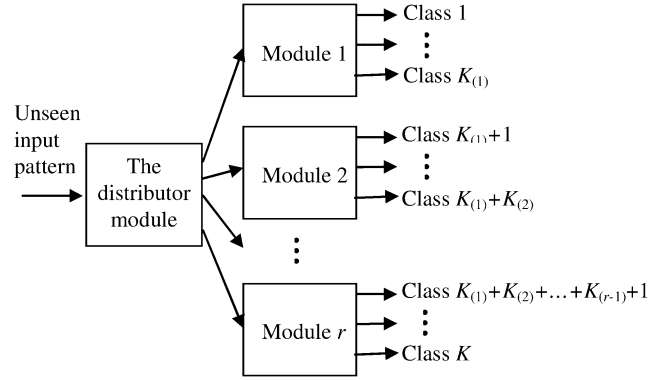


Fig. 2. PD network used to solve a K -class problem.

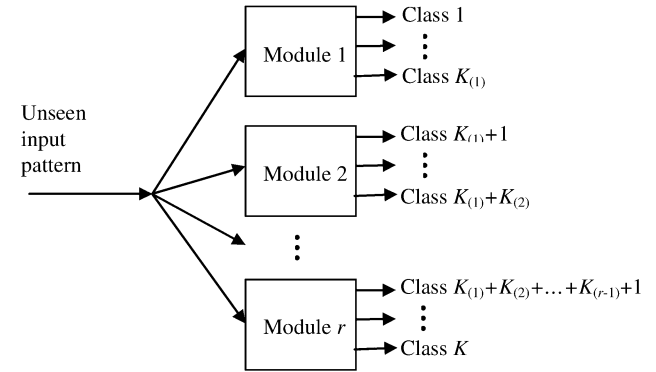


Fig. 3. OP network used for the same K -class problem.

outputs. Then, the connected nondistributor module continues the classification process to specify which class the pattern belongs to. In other words, a nondistributor module needs to recognize the pattern among several classes. Assume module j is a nondistributor module that needs to recognize $K_{(j)}$ classes. Different nondistributor modules are assumed to have no overlapping classes; we have the following:

$$K = \sum_{j=1}^r K_{(j)}. \quad (1)$$

Fig. 3 shows the corresponding OP network used to solve the aforementioned K -class problem. For the convenience of comparison, we assume that the OP network has the same output grouping as the PD network. There are also r modules in the OP network and module j needs to recognize $K_{(j)}$ classes among all the patterns. When an unknown input pattern is presented to the OP network, it is processed by each module (module 1 to module r), and the final result is obtained by integrating the results from module 1 to module r .

In the PD network, a nondistributor module only recognizes the patterns dispatched to it by the distributor module. These patterns most likely belong to one of the classes covered by that module. Of course, the distributor module may make wrong decisions and send wrong patterns to that module. The OP network is different. Each module needs to recognize all the patterns. In other words, module j in the OP network needs to differentiate the patterns belonging to it from those patterns which do not. Now, we denote the probability of error incurred by module

j processing the patterns that belong to one of the classes of module i by p_{ij} . If we do not implement winner-take-all arbitration, a pattern can be regarded as wrongly classified if one or more modules give wrong decisions. When a test pattern belonging to one of the classes of module j enters the network, the probability of error in the OP network can be written in the following form:

$$\begin{aligned}
p_j^{(\text{OP})} &= p_{1j} \prod_{\substack{i=1 \\ i \neq 1}}^r (1 - p_{ij}) + p_{2j} \prod_{\substack{i=1 \\ i \neq 2}}^r (1 - p_{ij}) + \cdots \\
&+ p_{rj} \prod_{\substack{i=1 \\ i \neq r}}^r (1 - p_{ij}) + p_{1j} p_{2j} \prod_{\substack{i=1 \\ i \neq 1,2}}^r (1 - p_{ij}) \\
&+ p_{1j} p_{3j} \prod_{\substack{i=1 \\ i \neq 1,3}}^r (1 - p_{ij}) + \cdots \\
&+ p_{(r-1)j} p_{rj} \prod_{\substack{i=1 \\ i \neq r-1,r}}^r (1 - p_{ij}) + \cdots \\
&+ p_{1j} p_{2j} \cdots p_{(r-1)j} p_{rj}. \tag{2}
\end{aligned}$$

The first r terms represent the probability of a test pattern being classified wrongly by one module. The following $(1)/(2)r \cdot (r-1)$ terms represent the probability of the test pattern being classified wrongly by two modules, and so on. Equation (2) can be rewritten as

$$\begin{aligned}
p_j^{(\text{OP})} &= p_{1j} + p_{2j} + \cdots + p_{rj} \\
&- (p_{1j} p_{2j} + p_{1j} p_{3j} + \cdots + p_{(r-1)j} p_{rj}) \\
&+ (p_{1j} p_{2j} p_{3j} + \cdots + p_{(r-2)j} p_{(r-1)j} p_{rj}) - \cdots \\
&+ (-1)^{r-1} (p_{1j} p_{2j} \cdots p_{rj}). \tag{3}
\end{aligned}$$

Here, r is the number of modules and is normally not a large number, so the number of terms in (3) is not a large number. p_{ij} is a small positive real number. In other words, $p_{ij} p_{kj}$ is much smaller than p_{ij} . We can ignore the terms of the product of two and more p_{ij} 's. Thus

$$p_j^{(\text{OP})} \approx p_{1j} + p_{2j} + \cdots + p_{rj}. \tag{4}$$

The number of test patterns classified wrongly by the OP network is

$$\begin{aligned}
N^{(\text{OP})} &= \sum_{j=1}^r N_j \cdot p_j^{(\text{OP})} = \sum_{j=1}^r N_j (p_{1j} + p_{2j} + \cdots + p_{rj}) \\
&= \sum_{j=1}^r \left[N_j \sum_{k=1}^r p_{kj} \right] \tag{5}
\end{aligned}$$

where N_j is the number of patterns belonging to the classes of module j . It can also be written as

$$\begin{aligned}
N^{(\text{OP})} &= \sum_{k=1}^r (N_1 p_{k1} + N_2 p_{k2} + \cdots \\
&+ N_k p_{kk} + \cdots + N_r p_{kr}). \tag{6}
\end{aligned}$$

Now, we define p_{k*} as the probability of error when module k processes the patterns not belonging to the classes of module k . Equation (6) can be revised as

$$N^{(\text{OP})} = \sum_{k=1}^r (N_k p_{kk} + (N - N_k) \cdot p_{k*}) \tag{7}$$

where p_{kk} is the probability of error when module k processes the patterns belonging to it, N is the number of test patterns, and N_k is the number of patterns belonging to the classes of module k .

It should be mentioned that, in the aforementioned OP network, each module can be trained separately using all the training patterns, whereas for the PD network, we can also train these modules separately. If we use all the training patterns to train these modules, then the weights and hidden units of the nondistributor modules will be the same as those of the corresponding modules in the OP network. After the training of the PD network is completed, the distributor module will be the first to classify any unseen input pattern. The corresponding output unit in the pattern distributor will have the largest output value among all the output units. Then, only the corresponding module will be activated. After that, the input pattern is presented to this module only, and then, this module will complete the classification process. Only the distributor and the corresponding modules are used in the classification process.

Let p_0 be the probability of error of the distributor module. Then, the number of test patterns which are classified wrongly by the distributor module is

$$M_0 = N \cdot p_0. \tag{8}$$

Assume the distributor module classifies patterns wrongly in a uniform manner. In other words, the number of wrongly classified patterns by the distributor module to each nondistributor module is proportional to the number of patterns entering that nondistributor module. The number of correct patterns that enter module j is $N_j(1 - p_0)$. Then, the number of patterns classified wrongly by module j is written as

$$M_j = N_j(1 - p_0)p_{jj}. \tag{9}$$

Thus, the number of patterns classified wrongly by the PD network can be expressed as

$$M^{(\text{PD})} = \sum_{i=0}^r M_i = N \cdot p_0 + (1 - p_0) \sum_{j=1}^r N_j \cdot p_{jj}. \tag{10}$$

Comparing the OP network with the PD network, we have

$$\begin{aligned}
N^{(\text{OP})} - M^{(\text{PD})} &= \sum_{j=1}^r [N_j p_{jj} + (N - N_j) p_{j*}] \\
&- N \cdot p_0 - (1 - p_0) \sum_{j=1}^r N_j \cdot p_{jj} \\
&= \sum_{j=1}^r (N - N_j) p_{j*} - N \cdot p_0 \\
&+ p_0 \sum_{j=1}^r N_j \cdot p_{jj}. \tag{11}
\end{aligned}$$

Similar to the analysis made earlier, $p_{j0} p_{jj}$ is much smaller than p_{j*} and p_0 , so

$$N^{(\text{OP})} - M^{(\text{PD})} \approx \sum_{j=1}^r (N - N_j) p_{j*} - N \cdot p_0. \tag{12}$$

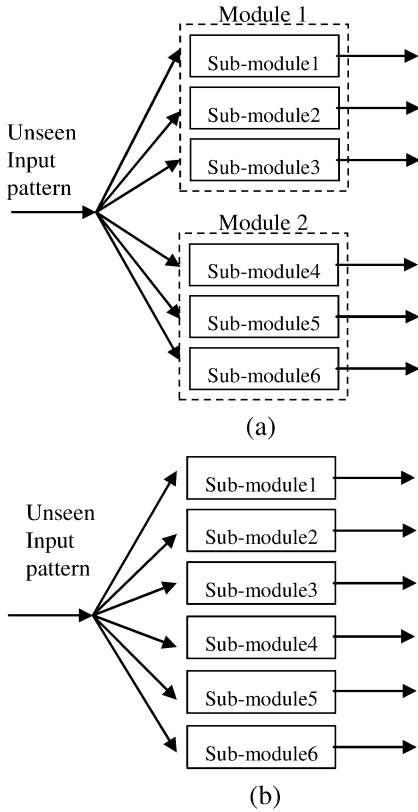


Fig. 4. Two OP networks for a six-class problem.

Now, we derive the condition under which the PD network can achieve better classification accuracy than the OP network

$$\sum_{j=1}^r (N - N_j) p_{j*} > N \cdot p_0. \quad (13)$$

We know that each module needs to process all the test patterns in an OP network, while in a PD network each nondistributor module only needs to process a subset of the test patterns. Intuitively, if the number of wrongly classified patterns by the distributor module in the PD network is smaller than the sum of the number of patterns wrongly classified by each module when processing patterns not belonging to it in the corresponding OP network, the PD network will perform better.

Discussions

- 1) Class decomposition can still be applied to the modules of the OP and PD networks so that these modules can be further decomposed into submodules. If each submodule is used to recognize one class from all the patterns, then there will be N submodules in the whole OP network. Of course, these submodules may belong to different modules. Fig. 4(a) shows an example of a six-class OP network. There are two modules that are further partitioned into six submodules. Fig. 4(b) shows a fully decomposed OP network for this six-class problem. In both OP networks, all the training patterns are used to train these submodules, so the submodules in Fig. 4(a) are the same as their counterparts in Fig. 4(b). In Fig. 4(a), the submodules

are grouped into two modules. For an unknown pattern, the outputs from submodules 1–3 are considered together to give the result of module 1, similar for module 2. Then, the results from modules 1 and 2 are considered together to give the final output. In the OP network of Fig. 4(b), the outputs from all the submodules are considered altogether to give the final output. In fact, there is little difference between the OP networks in Fig. 4(a) and (b). Note that the nondistributor modules in the PD network are the same as the counterparts in the OP network. Thus, by decomposing the modules into submodules, we can compare the performance of the PD network with that of the fully decomposed OP network. In most of our experiments, we used such networks.

- 2) In (4), we have ignored the situation in which two or more modules make wrong decisions at the same time because the situation appears much less frequently compared to the situation in which only one module makes wrong decisions. If we consider that situation, $p_j^{(OP)}$ will be a little smaller than $p_{1j} + p_{2j} + \dots + p_{rj}$.
- 3) In the previous model, we do not consider the implementation of winner-take-all for the OP network. In reality, winner-take-all is used for selecting a unit among several candidate units to produce the final output. The purpose of a conventional winner-take-all network is to select a unit with the highest activation strength from a set of candidates. Using winner-take-all, the network may still choose the correct output even if some modules make wrong decisions. For example, consider a test pattern that belongs to class A in module 1. When the pattern enters module 1 of the OP network, module 1 produces the correct answer—class A . However, when the pattern enters module 2 of the OP network, module 2 gives an incorrect answer and thinks it belongs to class B . If the output corresponding to class A is larger than that of class B , the OP network can still give a correct decision. Using winner-take-all will slightly reduce the final classification error of the OP network than not using it.

III. MOTIVATION FOR RPT

In a PD network, an unseen pattern is first classified by the distributor module to decide which module will continue to process it. Then, the corresponding module will be activated. Thus, only two modules are used to process that pattern. Now, we look at all the test patterns. We note that each nondistributor module only processes a subset of the test patterns. In other words, each nondistributor module only needs to recognize the patterns belonging to it if the distributor module classifies all the test patterns correctly. Also, if the distributor module classifies some patterns wrongly, the mistake cannot be corrected by the later modules. This motivates us to train a nondistributor module using only the patterns belonging to it. Such a method is called RPT. Similarly, the method of using the whole training set to train each nondistributor module is called full pattern training (FPT).

When we train module j using FPT, the module will carry information of the instances that do not belong to its own classes.

Such information does not contribute to the classification accuracy of module j , so it is useless. Also, training time would be reduced when training using RPT compared with FPT.

Moreover, training module j together with unnecessary patterns may reduce the ability of module j to classify the patterns belonging to module j correctly. There are two aspects. First, the objective of training is to let each module reach its best classification accuracy when processing the patterns dispatched to it. Using FPT, a module may be able to attain its best performance when it needs to process all the test instances. However, it may not attain its best performance when processing only a subset of the test instances. Second, for patterns not belonging to module j , it would have the outputs as 0 during the learning process. (In our experiments, if a pattern belongs to some class, the corresponding output is 1, otherwise, 0.) With the introduction of those patterns not belonging to module j , there are many more patterns with an output label 0 than patterns with an output label 1 in the learning process, so the patterns with an output label 0 will be more influential in updating the weights and, therefore, in computing the training error function. In contrast, the patterns with an output label 1 will become less influential in the decision of weight updates. After the training process is over, it is likely that the trained network may mislabel some test patterns, in particular those patterns with an output label 1. From the previous observations, we conclude those unnecessary patterns are harmful to the module training. Our experimental results confirmed that RPT is crucial for a PD network to obtain good performance.

RPT might not be applicable to OP, because the modules in an OP network operate in parallel and each module must deal with all the test patterns in the test process. Training these modules using reduced patterns may lead to information loss and to poor accuracy when the test patterns are presented.

IV. COMBINATION OF CLASSES IN THE DISTRIBUTOR MODULE

From the analysis in Section II, it can be seen that the performance of a PD network depends greatly on the accuracy of the distributor module. How to group the classes and combine them becomes a key issue in designing a PD network.

We define two concepts—combination and combination set. If some classes are grouped together, we call them a combination. The combination of classes A , B , and C is denoted as $\{A, B, C\}$. Once some classes are combined, they will form a new class. A combination set is an aggregation of combinations where each class in the original problem appears once and exactly once. For example, in a six-class problem, $\{\{1, 2, 3\}, \{4, 5\}, \{6\}\}$ is a combination set. Here, we present two methods to find an appropriate combination set in the distributor module.

A. Crosstalk-Based Combination

The basic idea is to find classes which are close in the feature space and combine them together. We first project a d -dimensional (d is the number of the input classes) input space to a 1-D space using Fisher's linear discriminant (FLD) method [12]. The distances between the centers of different classes are calculated. Then, these distances are arranged to form a table which is called crosstalk table. If the distance between two classes in the

crosstalk table is relatively small, then the two classes are likely to be close in the feature space. Thus, we choose and combine those classes that have relatively smaller distances from each other in the crosstalk table.

B. GA-Based Combination

The basic idea of this method is to find an optimal or near-optimal combination set through evolution. First, we define our chromosome encoding. A binary string of specific length is often used to encode a chromosome in canonical GAs, but it is not suitable here. Thus, we define chromosome according to the following principles. A chromosome consists of a sequence of combination numbers, wherein each class is encoded with its combination number. The length of a chromosome is equal to the number of the classes. Assume chromosome encoding always starts with the smallest class number and increases as follows. For example, 122333 is a chromosome for a six-class problem. Number "1" in the first place means class 1 belongs to combination 1. Similarly, number "3" in the fourth place means class 4 belongs to combination 3. The corresponding combination set of this chromosome is $\{\{1\}, \{2, 3\}, \{4, 5, 6\}\}$. There is a need for normalization, however. Let us look at another example—chromosome 233111. It is obvious that chromosomes 233111 and 122333 represent the same combination set (though the ordering differs). Therefore, chromosome 233111 can be normalized as 122333.

For convenience, we convert all the chromosomes into a form such as 122333. This process is called standardizing the chromosomes. The procedure of standardizing a chromosome is shown in the Appendix.

Then, we create an initial population of chromosomes. After generating the initial population, each chromosome is evaluated and assigned a fitness value. Here, we use a simple NN for the evaluation and use the classification error f of the validation data set to calculate the fitness

$$\text{fitness} = 2 - \frac{f}{f_{\text{ave}}} \quad (14)$$

where f_{ave} is the average of classification errors based on the validation data set for all the chromosomes in the population. f is also called evaluation value. If $2 - (f)/(f_{\text{ave}})$ is smaller than 0, $\text{fitness} = 0$.

The execution of our GA can be viewed as a two-stage process. It starts with the current population. Then, selection is applied to the current population to generate an intermediate population. After that, mutation and crossover are applied to the intermediate population to create the next population. We use "stochastic universal sampling" to form the intermediate population [11]. Assume that the population is laid out in random order as in a pie graph in which each individual is assigned space on the pie graph in proportion to *fitness*. Next, an outer roulette wheel is placed around the pie with N equally spaced pointers (N is the number of the population). A single spin of the roulette wheel will now simultaneously pick all N members of the intermediate population.

After the construction of the intermediate population, crossover and mutation are used to generate the next population. Crossover is applied to randomly paired chromosomes

with a probability p_c . Consider two chromosomes: 112233 and 122123. The random crossover point is chosen, for example, after the fourth place. Then, the numbers in the fifth and sixth places are exchanged and new chromosomes are formed. Here, the new chromosomes are 112223 and 122133. After crossover, mutation is applied to random chromosomes with a probability p_m . After a chromosome is selected for mutation, a place is randomly selected for mutation and the number in that place is randomly chosen. After the crossover and mutation is complete, standardize the chromosomes. Then, the next population is evaluated and becomes the current population. Then, the previously described process is repeated.

There is another important parameter N_{\max} —maximum number of classes in a combination. Using this parameter, we kick out some chromosomes directly. For a six-class problem, if we choose $N_{\max} = 3$, then chromosome 121112 will be eliminated, because combination $\{1, 3, 4, 5\}$ has four classes. The purpose of setting N_{\max} is to avoid the existence of nondistributor modules with many classes. If there are many classes in a nondistributor module, it is unlikely that this module can recognize patterns with a high classification rate. With a chromosome such as 111111, there is only one grouping and the job of the distributor would be trivial. For this extreme case, the classification error of the distributor module is obviously 0 because the distributor module combines all the classes together. Such an extreme case can be avoided using the parameter N_{\max} .

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Electronic Image Files (Optional)

Constructive backpropagation (CBP) algorithm was used to train the network in the experiments. Please refer to [13] for details. CBP can reduce the excessive computational cost significantly and it does not require any prior knowledge concerning decomposition. In this paper, RPROP¹ is used with the following parameters: $\eta^+ = 1.2, \eta^- = 0.5, \Delta_0 = 0.1, \Delta_{\max} = 50$, and $\Delta_{\min} = 1.0e - 6$ (η^+/η^- is the increase/decrease parameter, Δ_0 is the initial update value, and $\Delta_{\max}/\Delta_{\min}$ stands for the upper/lower limit of the update value) with initial weights selected from $-0.25 \dots 0.25$ randomly. Please refer to [14] for details. In order to avoid large computational cost and overfitting, a method called early stopping based on validation set is used as the stopping criteria. Please refer to [22] for details.

The set of available patterns is divided into three sets: a training set is used to train the network, a validation set is used to evaluate the quality of the network during training and to measure overfitting, and a test set is used at the end of training to evaluate the resultant network. The size of the training, validation and test sets is 50%, 25%, and 25% of the problem’s total available patterns.

Four benchmark classification problems, namely *vowel*, *glass*, *segmentation*, and *letter recognition* were used to evaluate the performance of the new modular network—*task*

¹RPROP stands for “resilient propagation,” which is a learning algorithm that performs a direct weight update based on local gradient information (refer to [14]).

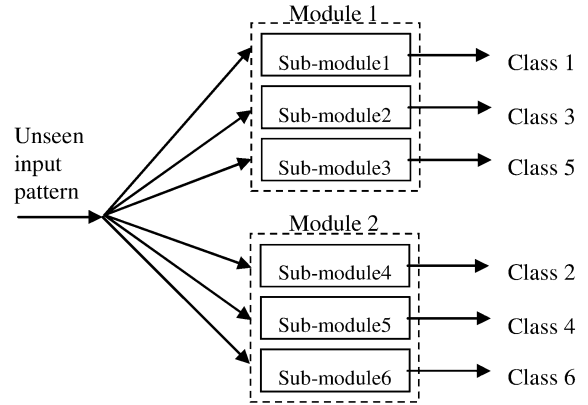


Fig. 5. OP network used for the glass problem.

TABLE I
CLASSIFICATION ERROR IN DIFFERENT OP MODULES FOR THE GLASS DATA

Output Parallelism	N_i	p_{ii} (%)	$N - N_i$	p_{i^*} (%)
Module 1	67	8.4142	94	4.7340
Module 2	94	18.1383	67	2.1642

decomposition with PD. These classification problems were taken from the PROBEN1 benchmark collection [15] and the University of California at Irvine (UCI) repository of machine learning database [16]. In the set of experiments undertaken, the first three classification problems were conducted 20 times and the *letter recognition* problem was conducted eight times (due to the long training time). All the hidden units and output units use the sigmoid activation function and E_{th} is set at 0.1. When a hidden unit addition was required, eight candidates were trained and the best one selected. All the experiments were simulated on a Pentium IV 2.4-GHZ PC. The subproblems were solved sequentially and the central processing unit (CPU) time expended was recorded, respectively.

B. Experiments for PD Network Based on FPT and RPT

1) *Glass*: This data set is used to classify glass types. The data set consists of nine inputs, six outputs, and 643 patterns (divided into 321 training patterns, 161 validation patterns, and 161 test patterns). These patterns are normalized and scaled so that each component lies within $[0, 1]$.

Fig. 5 shows the OP network structure used for this problem. The OP network is composed of six submodules and each submodule recognizes one class from all the patterns. As described in Discussion 1 in Section II, these submodules are combined into two modules in the OP network. The submodules which recognize classes 1, 3, and 5 are combined into module 1 and the remaining submodules are grouped into module 2.

Table I lists some data which are used in (13). Here, N_i represents the number of the patterns in the test data set belonging to the classes of module i while N denotes the overall number of the patterns. p_{ii} is the probability of error when module i processes the patterns belonging to module i and p_{i^*} is the probability of error when module i processes the

TABLE II
RESULTS FOR THE GLASS DATA

Method	Training time (s)	Hidden Units	Indep. Param.	C. error (%)
Ordinary method (no task decomposition)	168.1	46	796	16.0870
Output Parallelism (2 modules, 6 sub-modules)	63.7 (parallel)	253.5	2848.5	14.2547
	197.7 (series)			
The distributor module	82.9	30.6	387.2	2.4224
Pattern Distributor (1 distributor module, 2 modules,	Overall network (FPT)	280.9	3200.5	10.0
	85.2 (parallel)			
6 sub-modules)	Overall network (RPT)	391.2	4413.8	7.8261
	82.9 (parallel)			
	194.3 (series)			

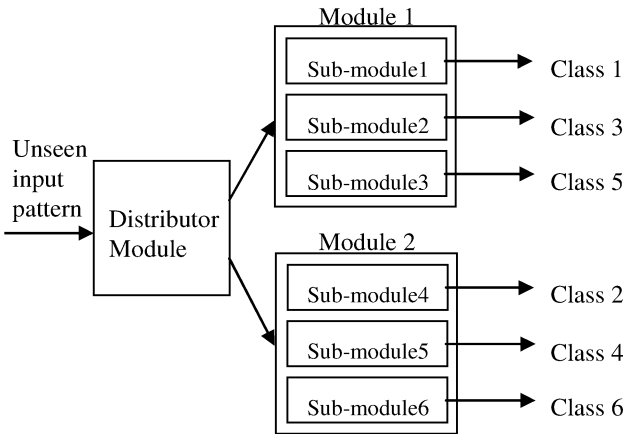


Fig. 6. PD network used for the glass problem.

patterns not belonging to module i . Now, we show that Discussion 2 in Section II is reasonable. There are two modules in the OP network. From Table I, we have $p_{11} = 8.4142\%$ and $p_{12} = p_{1^*} = 4.7340\%$. So $p_{11} \cdot p_{12} = 0.40\%$, which is much smaller than p_{11} and p_{12} . It is similar that $p_{22} \cdot p_{21} = 0.39\%$, which is much smaller than p_{21} and p_{22} . Ignoring these terms has little effect to the final results, in other words, the situation in which two or more modules making wrong decisions at the same time can be ignored. Now, we follow up on the Discussion 3 in Section II—the effect of winner-take-tall. From Table I, we can compute the classification error before the implementation of winner-take-all, which is $N_1 \cdot p_{11} + N_2 \cdot p_{1^*} + N_2 \cdot p_{22} + N_1 \cdot p_{2^*} = 17.7562\%$. The result is slightly larger than the result using winner-take-all, which is 14.2547% (see Table II). It also matches our analysis in Discussion 3 in Section II.

The PD network structure for this problem is shown in Fig. 6. The distributor module has two outputs; one has the combination $\{1, 3, 5\}$ while the other has $\{2, 4, 6\}$. Module 1 consists of three submodules, identical to its counterpart in the OP network, and the same for module 2.

Table II shows the experimental results of the ordinary method, the OP method, the PD method with FPT, and the PD method with RPT. The ordinary method is a method in which a single-module NN was constructed to solve the problem. CBP algorithm is still used in the ordinary method. “Indep. Param.” stands for the total number of independent parameters (i.e., the number of weights and biases in the network). “C. Error” stands for classification error. Training time (in parallel) is the maximum training time among all the modules (all modules were trained in parallel). Training time (in series) stands for the sum of training time for all the modules (all modules were trained in series). Using the ordinary and the OP methods, the classification errors were 16.0870% and 14.2547%, respectively, while using the PD method, the classification errors were 10% for FPT and 7.8261% for RPT. Comparing with the classification errors from the former two approaches, the classification errors obtained by the PD network are much smaller. It can be also noted that the classification error is further reduced when using RPT instead of FPT.

Now, we explain why the PD network can achieve smaller classification error than the other two methods. According to our analysis, if (13) is satisfied, the PD network will have better classification accuracy. Using the data in Table I, we have $\sum_{j=1}^2 (N - N_j) p_{j^*} \approx 5.9$. From the classification error p_0 of the distributor module in Table II, we find $N \cdot p_0 \approx 3.9$. Thus, (13) is satisfied, which means that using the PD network will get smaller classification error. From Table II, we can see that the number of hidden units and the number of independent parameters in the PD network are larger than those in the ordinary network and the OP network. This can be attributed to the fact that the PD network has more modules than the other two. From Table II, we can also note the changes of the training time using the previously described three methods. With series training, the training time of FPT (298.7 s) is larger than those of the ordinary network (168.1 s) and the OP network (197.7 s) due to a large number of modules in the PD method. However, the training time of RPT (194 s) is reduced compared to that of FPT and is thus comparable to the training time of the other two networks. The reason for this is that the number of training instances used in RPT is smaller than that in FPT. With parallel training, the training time of the PD network (RPT or FPT) is similar to those of the other two methods, and it is even shorter than that of the ordinary method. From the previous analysis, we see that the PD method, especially RPT, performs better than the other methods.

2) *Vowel*: The input patterns of this data set are ten element real vectors representing vowel sounds that belong to one of 11 classes. It has 990 patterns in total (they are divided into 495 training patterns, 248 validation patterns, and 247 test patterns). The patterns are normalized and scaled so that each component lies within $[0, 1]$. The distributor module has three outputs $\{1, 2, 3\}$, $\{4, 5, 6, 7\}$, and $\{8, 9, 10, 11\}$. Module 1 recognizes classes 1–3 and consists of three submodules. Module 2 recognizes classes 4–7 and consists of four submodules, while module 3 recognizes classes 8–11 and consists of four submodules. The OP network has the same modules 1, 2, and 3 as the PD network.

The experimental results of the ordinary, the OP, and the PD methods for the vowel data are listed in Table IV. Using the

TABLE III
CLASSIFICATION ERROR IN DIFFERENT OP MODULES FOR THE VOWEL DATA

Output Parallelism	N_i	p_{ii} (%)	$N-N_i$	p_{i^*} (%)
Module 1	69	9.8551	178	2.9775
Module 2	96	32.0833	151	3.2450
Module 3	82	34.7561	165	5.8788

TABLE IV
RESULTS FOR THE VOWEL DATA

Method	Training time (s)	Hidden Units	Indep. Param.	C. error (%)
Ordinary method (no task decomposition)	237.9	23.6	640.2	37.1660
Output Parallelism (3 modules, 11 sub-modules)	58.7 (parallel) 418.9 (series)	184.4	2333.8	25.5466
The distributor module	117	24.5	376	6.6802
Pattern Distributor (1 distributor module, 3 modules and 11 sub-modules)	Overall network (FPT) 534.3 (series) Overall network (RPT) 245.6 (series)	210.6 (parallel)	2730.2	24.8987
	117 (parallel) 229.4 (series)	229.4	2955.8	18.7045

TABLE V
CLASSIFICATION ERROR IN DIFFERENT OP MODULES FOR THE SEGMENTATION DATA

Output Parallelism	N_i	p_{ii} (%)	$N-N_i$	p_{i^*} (%)
Module 1	246	10.4129	331	0.2417
Module 2	331	0.9215	246	0.6098

TABLE VI
RESULTS FOR THE SEGMENTATION DATA

Method	Training time (s)	Hidden Units	Indep. Param.	C. error (%)
Ordinary method (no task decomposition)	693.8	29	887	5.7366
Output Parallelism (7 sub-modules)	610.2 (parallel) 1719.6 (series)	152.1	3175	5.1820
The distributor module	213.4	13.9	329.9	1.0399
Pattern Distributor (1 distributor module, 2 modules and 7 sub-modules)	Overall network (FPT) 1002.2 (parallel) 2219.2 (series) Overall network (RPT) 706.9 (series)	128.5	2754.9	5.4419
	213.4 (parallel) 706.9 (series)	128.9	2762.9	4.6101

ordinary and the OP methods, the classification errors were 37.1660% for the ordinary method and 25.5466% for the OP method, respectively, while using the PD method, the classification errors were 24.8987% for FPT and 18.7045% for RPT. The classification error obtained by FPT is much smaller than the classification error of the ordinary method and resembles that of the OP method, while for RPT, the classification error is decreased to 18.7045%, which is much smaller than those of FPT and the other two methods. We can compute the number of wrongly classified patterns using the data in Table III to explain why the PD method can get smaller classification errors than the other two methods. We have $\sum_{j=1}^3(N - N_j)p_{j^*} \approx 19.9$ while $N \cdot p_0 \approx 16.5$. Expression (13) is satisfied. Thus, the PD network has smaller classification errors. From Table IV, we can see that the number of hidden units and the number of independent parameters in the PD network (RPT or FPT) are larger than those in the ordinary and OP networks. Table IV also shows the training time using these methods. Using series training, the training time of FPT (534.3 s) is longer than those of the ordinary network (237.9 s) and the OP network (418.9 s). The training time of RPT (245.6 s) is much reduced compared to that of FPT and is also smaller than those of the former two networks. If parallel training is used, the training process of the PD network can save more time. From the previous analysis, we see that RPT outperforms the others.

3) *Segmentation*: This data set consists of 18 inputs, seven outputs, and 2310 patterns (1155 training patterns, 578 validation patterns, and 577 test patterns). The patterns are normalized and scaled so that each component lies within [0, 1]. The

distributor module has two outputs {3, 4, 5} and {1, 2, 6, 7}. Module 1 recognizes classes 3–5 and consists of three submodules. Module 2 recognizes classes 1, 2, 6, and 7 and consists of four submodules. The OP network has the same module composition as the PD network.

Table VI shows the simulation results of the ordinary method, the OP method, the PD method (FPT and RPT). Using the ordinary method and the OP method, the classification errors were 5.7366% and 5.1820%, respectively, while using the PD method, the classification errors were 5.4419% for FPT and 4.6101% for RPT. From Table V, we have $\sum_{j=1}^2(N - N_j)p_{j^*} \approx 2.3$. From Table VI, we find $N \cdot p_0 \approx 6.0$, so (13) is not satisfied and FPT has a larger classification error than the OP network. It is also noted that the classification error is decreased when using RPT to replace FPT. From Table VI, we can see that the number of hidden units and the number of independent parameters in the PD networks are larger than those in the ordinary and OP networks. From Table VI, we also notice changes in training time using the previous three methods. Under series training, the training time of FPT (2219.2 s) is larger than the training times of the ordinary network (693.8 s) and the OP network (1719.6 s) due to a large number of modules in the PD network. However, the training time of RPT (706.9 s) is reduced compared to that of FPT and the OP network and is thus comparable to the training time of the ordinary method. With parallel training, the training time of RPT is the smallest one. From the previous analysis, we see that RPT performs better than the other methods.

TABLE VII
CLASSIFICATION ERROR IN DIFFERENT OP MODULES FOR THE LETTER DATA

Output Parallelism	N_i	p_{ii} (%)	$N-N_i$	p_{i^*} (%)
Module 1	1359	20.833	3641	2.856
Module 2	1333	24.812	3667	1.084
Module 3	1195	25.109	3805	3.035
Module 4	1113	11.051	3889	1.826

TABLE VIII
RESULTS FOR THE LETTER DATA

Method	Training time (s)	Hidden Units	Indep. Param.	C. error (%)	
Ordinary method (no task decomposition)	20845.05	73.6	3607	21.672	
Output Parallelism (4 modules)	5519 (parallel) 18112.6 (series)	173.4	6586.8	19.260	
Pattern Distributor (1 distributor module, 4 modules)	The distributor module	2510 (parallel) 8497 (series)	219.5	4019.0	12.195
	Overall network (FPT)	6110 (parallel) 26723.8 (series)	386.25	8384.5	20.515
	Overall network (RPT)	2510 (parallel) 14094.5 (series)	344.25	7391.0	15.855

4) *Letter Recognition*: The goal of this data is to recognize digitized patterns. Each element of the input vector is a numerical attribute computed from a pixel array containing the letters. This data set consists of 16 inputs, 26 outputs, and 20000 patterns (10000 training patterns, 5000 validation patterns, and 5000 test patterns). All the patterns are normalized and scaled so that each component lies within $[0, 1]$. The distributor module has four outputs $\{1, 2, 3, 4, 5, 6, 7\}$, $\{8, 9, 10, 11, 12, 13, 14\}$, $\{15, 16, 17, 18, 19, 20\}$, and $\{21, 22, 23, 24, 25, 26\}$. Module 1 recognizes classes 1–7. Due to the long training time of this problem, module 1 is not further divided into submodules. Module 2 recognizes classes 8–14, module 3 recognizes classes 15–20, and module 4 recognizes classes 21–26. The OP network has the same module composition as the PD network. For a fair comparison with the PD network, submodules are not used in the OP network.

The experimental results of the ordinary method, the OP method and the PD method for the letter data are listed in Table VIII. Using the ordinary method and the OP method, the classification errors were 21.672% for the ordinary method and 19.260% for the OP method, respectively. Using the PD method, the classification error were 20.515% for FPT and 15.855% for RPT. The classification error obtained by FPT resembles the classification errors using the ordinary method and the OP method. Using RPT, the classification error is much

TABLE IX
CROSSTALK TABLE FOR THE SEGMENTATION DATA

	1	2	3	4	5	6	7
1	0	233.0	9.948	17.66	8.876	40.34	53.91
2	233.0	0	72.53	17.12	53.10	42.95	518.8
3	9.948	72.53	0	6.767	1.648	18.55	40.66
4	17.66	17.12	6.767	0	3.062	4.458	54.70
5	8.876	53.10	1.648	3.062	0	16.34	29.02
6	40.34	42.95	18.55	4.458	16.34	0	60.95
7	53.91	518.8	40.66	54.70	29.02	60.95	0

smaller than the classification errors of the other three networks. From Table VII, we have $\sum_{j=1}^2 (N - N_j)p_{j^*} \approx 330.3$. From Table VIII, we find $N \cdot p_0 \approx 609.8$, so (13) is not satisfied, which means that FPT has a larger classification error. From Table VIII, we see that the number of hidden units and the number of independent parameters in the PD network are larger than those in the ordinary and OP networks. Table VIII also shows the training time using these methods. Under series training, the training time of FPT (26723.8 s) is larger than those of the ordinary network (18112.6 s) and the OP network (20845.05 s). The training time of RPT (14094.5 s) is greatly decreased compared to that of FPT and is also smaller than those of the former two networks. If parallel training were used, the training process of RPT could have saved more time. From the previous analysis, we can see that RPT performs better than the others.

C. Crosstalk-Based Combination for Distributor Modules

Two classification problems, namely *segmentation* and *glass*, were used in the experiments.

1) *Segmentation*: The crosstalk table for this problem is obtained using the method described in Section IV-A, as shown in Table IX. As mentioned in Section IV-A, if the distance between two classes in the crosstalk table is relatively small, then these two classes are likely to be close in the feature space. From Table IX, we want to find classes which are close to each other and combine them. For simplicity, we use $d(i, j)$ to denote the distance between classes i and j . From the table, $d(3, 5)$ is 1.6476, $d(4, 5)$ is 3.0618, and $d(3, 4)$ is 6.7673. These distances are relatively small compared with other distance figures. Thus, we combine classes 3–5 together. In the remaining four classes, class 1 is relatively close to classes 3–5. Thus, we combine 1, 3, 4, and 5 together. Now, look at classes 2, 6, and 7. Classes 2 and 6 are relatively close and we combine them together. The final combination set is $\{\{1, 3, 4, 5\}, \{2, 6\}, \{7\}\}$.

We use another combination set $\{\{1, 2, 7\}, \{3, 4\}, \{5, 6\}\}$ for comparison with the previous set. In this set, we combine together the classes with relatively large distances. The experimental results for these two partitions are shown in Table X. Table X shows that the distributor module's classification error

TABLE X
RESULTS FOR THE SEGMENTATION PROBLEM
USING CROSSTALK-BASED COMBINATION

Grouping of Output classes	The distributor module's Classification error (%)	Overall Classification error (%)
Module 1 {1,3,4,5} Module 2{2,6} Module 3{7} (RPT)	0.1040	4.6187
Module 1{1,2,7} Module 2{3,4} Module 3{5,6} (RPT)	4.7834	5.3900

TABLE XI
CROSSTALK TABLE FOR THE GLASS DATA

	1	2	3	4	5	6
1	0	0.4467	0.3481	13.0807	7.7731	10.8269
2	0.4467	0	0.501	1.2606	1.9163	5.8246
3	0.3481	0.501	0	26.312	9.2086	7.6053
4	13.0807	1.2606	26.312	0	4.6975	6.8534
5	7.7731	1.9163	9.2086	4.6975	0	2.1657
6	10.8269	5.8246	7.6053	6.8534	2.1657	0

TABLE XII
RESULTS FOR THE GLASS PROBLEM USING CROSSTALK-BASED COMBINATION

Grouping of Output classes	The distributor module's Classification error (%)	Overall Classification error (%)
Module 1{1,2,3} Module 2{4,5,6} (RPT)	2.4224	7.5776
Module 1{3,4,6} Module 2{1,2,5} (RPT)	4.5963	8.5093

as well as the overall classification error are reduced when the classes close to each other are combined together.

2) *Glass*: The crosstalk table for this data set is shown in Table XI. We can see that the distances among classes 1–3 are 0.4467, 0.3481, and 0.501, which are much smaller than the other distances, so classes 1–3 are combined. In the remaining classes, it seems that class 4 is close to class 2. However, $d(4, 1)$ and $d(4, 3)$ are very large. Class 4 is not added to combination {1, 2, 3}. Note that classes 4–6 have relatively small distances. Thus, classes 4–6 are combined. Thus, the final combination set is $\{\{1, 2, 3\}, \{4, 5, 6\}\}$.

We use another combination set $\{\{3, 4, 6\}, \{1, 2, 5\}\}$ for comparison with the previous set. In this set, we combine together the classes with relatively large distances. The experimental results for the two different partitions are shown in Table XII. From Table XII, it is confirmed that the distributor module's classification error as well as the overall classification error are reduced when the classes close to each other are combined together.

TABLE XIII
RESULTS OF THE SEGMENTATION PROBLEM USING GA-BASED COMBINATION

Grouping of Output classes	The distributor module's Classification error (%)	Overall Classification error (%)
Module 1 {1,3,4,5} Module 2 {6,7} Module 3 {2} (RPT)	0.0173	4.5321

TABLE XIV
RESULTS OF THE GLASS PROBLEM USING GA-BASED COMBINATION

Grouping of Output classes	The distributor module's Classification error (%)	Overall Classification error (%)
Module 1 {1,3,5} Module 2{2,4,6} (RPT)	2.4224	7.8261

D. GA-Based Combination for Distributor Modules

To compare with the results in Section V-C, the same classification problems, namely *segmentation* and *glass*, were used in the experiments. In the experiments, we chose the probability of crossover $p_c = 0.2$ and the probability of mutation $p_m = 0.2$. For each chromosome, the classification error of the validation set is computed five times. The evaluation value is the average of the classification errors from five runs.

1) *Segmentation*: In the experiments, we set the maximum number of combination $N_{\max} = 4$. The population number is 20. Due to long computation time, only 30 generations were bred in our experiments. We identified the best chromosome 1211133, or $\{\{1, 3, 4, 5\}, \{6, 7\}, \{2\}\}$. The experimental results are shown in Table XIII. Comparing the results in Table XIII with those in Table XI, it can be seen that using GA-based combination, the classification error of the distributor module is decreased from 0.1040% to 0.0173%. The classification error of the whole network is slightly better than that using crosstalk-based combination.

2) *Glass*: In the experiments, we set the maximum number of combination $N_{\max} = 3$. The population number is 12. Due to long computation time, only 30 generations were bred in our experiments. We identified the best chromosome 121212, or $\{\{1, 3, 5\}, \{2, 4, 6\}\}$. The experimental results are shown in Table XIV. Comparing the results in Table XIV with those in Table XII, it can be seen that using GA-based combination, the classification error of the distributor module is equal to that using crosstalk-based combination. The classification error of the whole network is slightly larger than that using crosstalk-based combination.

In the previous two sets of experiments, it took 11 epochs for the glass problem and 14 epochs for the segmentation problem to locate the best chromosome. With the increasing number of classes, the number of epochs required to locate the best chromosome is also increased. In the previous two examples, we see that the classification error of the distributor module using GA-based combination seems better than or equal to that using crosstalk-based combination. However, the whole network's performance using GA-based combination is not always better than that using crosstalk-based combination. It

is also related to the recognition rates of the nondistributor modules. It can be seen that GA-based combination may not be a good choice compared with crosstalk-based combination in these two examples, due to the fact that the improvement in classification rate is trivial while much more computation is needed for GA-based combination. For problems with a large number of classes whose crosstalk computation is more costly and harder to analyze, GA-based combination may be a better choice. On the other hand, we may consider generating some initial chromosomes based on the crosstalk analysis to further improve the quality of GA-based combination.

VI. CONCLUSION

This paper presented a unique task decomposition approach called taskdecomposition with PD. In this design, a special module called distributor module was introduced in order to improve the accuracy of the whole network. A theoretical model was shown to compare the performance of PD with that of OP—a typical class decomposition method. The analysis showed that PD can outperform OP when the classification accuracy of the distributor module is guaranteed. The experimental results confirmed this. In order to further improve the performance of PD, RPT was introduced. RPT apparently increased the accuracy of the PD network. According to our model, the distributor module's classification accuracy dominated the whole network's performance. Two combination methods, crosstalk-based combination and GA-based combination, were proposed to find good class grouping for the distributor module. Crosstalk-based combination could find a suitable combination set for the distributor module. GA-based combination could find the optimal (or near-optimal) combination set for the distributor module, with a larger computation cost. Our experimental results confirmed the effectiveness of the combination methods proposed.

We will continue to improve the combination methods in the future. We hope to design new combination methods which not only can find optimal or near-optimal sets for the distributor module but also reduce further the computation time. In our paper, the number of distributor module is restricted to one. This can be relaxed by having multilevel PD networks with two or more distributor modules. How to reduce further the training pattern set while retaining the recognition rate will also be explored in our future research.

APPENDIX

The procedure of standardizing a chromosome is shown as follows.

- 1) Add a minus sign “-” to all the places. For example, a place with number “3” now becomes “-3.” Chromosome 233111 becomes $(-2)(-3)(-3)(-1)(-1)(-1)$.
- 2) Set $t = 1$. Find the number in the first place and find all the places with the same number as the first one. Change the numbers in the first place and all the matching places into “ t .” In the previous example, chromosome $(-2)(-3)(-3)(-1)(-1)(-1)$ becomes $1(-3)(-3)(-1)(-1)(-1)$.
- 3) Set $t = t + 1$. Scanning from left to right, find the leftmost place whose number is negative and

find all the following places whose number is the same. Change the numbers in these places into “ t .” In the previous example, when $t = 2$, chromosome $1(-3)(-3)(-1)(-1)(-1)$ becomes $122(-1)(-1)(-1)$.

- 4) Repeat step 3) until all the places have positive numbers inside.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments.

REFERENCES

- [1] R. A. Jacobs, M. I. Jordan, M. I. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.
- [2] S. U. Guan and S. C. Li, “Parallel growing and training of neural networks using output parallelism,” *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 542–550, May 2002.
- [3] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, “Efficient classification for multiclass problems using modular neural networks,” *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 117–124, Jan. 1995.
- [4] B. L. Lu and M. Ito, “Task decomposition and module combination based on class relations: A modular neural network for pattern classification,” *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1244–1256, Sep. 1999.
- [5] S. U. Guan and S. C. Li, “An approach to parallel growing and training of neural networks,” in *Proc. IEEE Int. Symp. Intell. Signal Process. Commun. Syst.*, Honolulu, HI, 2000, vol. 2, pp. 1101–1104.
- [6] S. U. Guan and J. Liu, “Feature selection for modular networks based on incremental training,” *J. Intell. Syst.*, vol. 13, pp. 45–70, 2004.
- [7] S. U. Guan and F. M. Zhu, “Modular feature selection using relative importance factors,” *Int. J. Comput. Intell. Appl.*, vol. 4, pp. 57–75, 2004.
- [8] S. U. Guan and F. M. Zhu, “Class decomposition for GA-based classifier agents—A Pitt approach,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 381–392, Feb. 2004.
- [9] S. U. Guan and R. Kiruthika, “Recursive percentage based hybrid pattern (RPHP) training for curve fitting,” in *Proc. IEEE Int. Conf. Cybern. Intell. Syst.*, 2004, pp. 445–450.
- [10] B. L. Lu, H. Kita, and Y. Nishikawa, “A multisieving neural-network architecture that decomposes learning tasks automatically,” in *Proc. IEEE Conf. Neural Netw.*, Orlando, FL, 1994, pp. 1319–1324.
- [11] J. Baker, “Reducing bias and inefficiency in the selection algorithm,” in *Proc. 2nd Int. Conf. Genetic Algorithms Appl.*, Cambridge, MA, 1987, pp. 14–21.
- [12] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Academic, 1973.
- [13] M. Lehtokangas, “Modeling with constructive backpropagation,” *Neural Netw.*, vol. 12, pp. 707–716, 1999.
- [14] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: The RPROP algorithm,” in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 586–591.
- [15] L. Prechelt, “PROBEN1: A set of neural network benchmark problems and benchmarking rules,” Dept. Inf., Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.
- [16] C. L. Blake and C. J. Merz, “UCI repository of machine learning databases,” Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998 [Online]. Available: <http://www.ics.uci.edu/~mllearn/MLRepository.html>
- [17] C. S. Squires and J. W. Shavlik, “Experimental analysis of aspects of the cascade-correlation learning architecture,” Dept. Comput. Sci., Univ. Wisconsin, Madison, WI, Mach. learn. Res. Group Working Paper 91-1, 1991.
- [18] L. Prechelt, “Investigation of the CasCor family of learning algorithms,” *Neural Netw.*, vol. 10, no. 5, pp. 885–896, 1997.
- [19] N. J. Nilsson, *Learning Machines: Foundations of Trainable Pattern Classifying Systems*. New York: McGraw-Hill, 1965.
- [20] M. R. Berthold and J. Diamond, “Constructive training of probabilistic neural networks,” *Neurocomputing*, vol. 19, pp. 167–183, 1998.
- [21] S. Ishihara and T. Nagano, “Text-independent speaker recognition utilizing neural-network techniques,” (in Japanese) *Tech. Rep. Inst. Electron. Inf. Commun. Eng. (IEICE)*, vol. NC93-121, pp. 71–77, 1994.

- [22] G. Auda, M. Kamel, and H. Raafat, "Modular neural network architectures for classification," in *Proc. IEEE Int. Conf. Neural Netw.*, 1996, vol. 2, pp. 1279–1284.
- [23] Y. Li, M. Dong, and R. Kothari, "Classifiability-based omnivariate decision trees," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1547–1560, Nov. 2005.
- [24] K. Z. Mao and G. B. Huang, "Neuron selection for RBF neural network classifier based on data structure preserving criterion," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1531–1540, Nov. 2005.
- [25] A. Szymkowiak-Have, M. A. Girolami, and J. Larsen, "Clustering via kernel decomposition," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 256–264, Jan. 2006.
- [26] N. Takahashi and T. Nishi, "Global convergence of decomposition learning methods for support vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1362–1369, Nov. 2006.



Sheng-Uei Guan received the M.Sc. and Ph.D. degrees from the University of North Carolina, Chapel Hill, in 1987 and 1989, respectively.

Currently, he is a Professor at the School of Engineering and Design, Brunel University, Uxbridge, Middlesex, U.K. He has worked in a prestigious R&D organization for several years, serving as a Design Engineer, Project Leader, and Manager. After leaving the industry, he joined Yuan-Ze University, Taiwan. He served as the Deputy Director for the Computing Center and the Chairman for the Department of In-

formation and Communication Technology. Later, he joined the Electrical and Computer Engineering Department, National University of Singapore as an Associate Professor.



Chunyu Bao received the B.Sc. and M.Sc. degrees in physics from Beijing University, Beijing, P.R. China, in 1999 and 2002, respectively. Currently, he is working towards the Ph.D. degree at the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.

His research interests include machine learning, neural networks, pattern classification, and clustering.



TseNgee Neo received the B.Eng. degree from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.