Task Decomposition Using Pattern Distributor

Sheng-Uei Guan*, TseNgee Neo and Chunyu Bao

Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260
*sg_1_1@yahoo.com/eleguans@nus.edu.sg

Abstract — In this paper, we propose a new task decomposition method for multilayered feedforward neural networks, namely *Task Decomposition with Pattern Distributor* in order to shorten the training time and improve the generalization accuracy of a network under training. This new method uses the combination of modules (small-size feedforward network) in parallel and series, to produce the overall solution for a complex problem. Based on a "divide-and-conquer" technique, the original problem is decomposed into several simpler sub-problems by a *pattern distributor* module in the network, where each sub-problem is composed of the whole input vector and a fraction of the output vector of the original problem. These sub-problems are then solved by the corresponding groups of modules, where each group of modules is connected in series with the pattern distributor module and the modules in each group are connected in parallel. The design details and implementation of this new method are introduced in this paper. Several benchmark classification problems are used to test this new method. The analysis and experimental results show that this new method could reduce training time and improve generalization accuracy.

Keywords — Task decomposition, multilayered feedforward neural network, pattern distributor

1. Introduction

Multilayered feedforward neural networks have been used extensively in solving classification problems. However, the concomitant disadvantages of building multilayered feedforward networks are the long training time and unsatisfactory generalization accuracy. One of the main reasons that cause these disadvantages is because large networks tend to introduce high internal interference due to the strong coupling among their hidden-layer weights (Jacobs et al., 1991). During the weight-updating (training) process, the influences (desired outputs) from two or more output units could cause the hidden-layer weights to compromise to non-optimal values due to the interference in their weight-updating direction. In order to overcome this drawback, various task decomposition methods based on "divide-and-conquer" have been proposed. Instead of using a single, large feedforward network (classic non-modular network), these task decomposition methods use a modular network, which is formed by integrating several modules (each module is a small size feedforward network) to solve the given problem. In the following section, several task decomposition methods are discussed.

2. Task decomposition methods

The method proposed in "Efficient classification for multiclass problems using modular neural networks" presented by Anand, etc. in 1995 divides a K-class original problem into K two-class sub-problems and each sub-problem is solved by a single-output module (small size feedforward network) respectively. Therefore, each module is used to discriminate one class of patterns from patterns belonging to the remaining classes. The collection of all the modules produces the overall solution for the original problem. Another method proposed in "Task decomposition and module combination based on class relations: A modular neural network for

pattern classification" splits the K-class original problem into $\binom{K}{2}$ two-class sub-problems. Each sub-problem is learned independently by a module while training patterns belonging to the other $K-2$ classes is ignored (Lu and Ito, 1999). The final overall solution is obtained by integrating all of the trained modules into a min-max modular network. The output parallelism method decomposes the original complex problem into a set of simpler sub-problems without any prior knowledge concerning the decomposition of the problem (Guan and Li, 2000 and Guan and Li 2002). Each sub-problem is composed of the whole input problem space and a fraction of the output problem space as illustrated in figure 1:

"Figure 1 near here"
"Figure 2 near here"

Each sub-problem is then solved by building and training a module. A collection of these modules (in parallel) is the overall solution of the original problem. The overview of the final network architecture is illustrated in figure 2.

Instead of decomposing the problem with high dimensional output space into several sub-problems with low dimensional output space, the method proposed in "A multisieving neural-network architecture that decomposes learning tasks automatically" by Lu et al. in 1994 decomposes the size (number of patterns) of the problem into several smaller size sub-problems. Patterns are classified by a rough sieve module (non-modular 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 into the network adaptively with progress of training.

The training time of these methods is shorter and the generalization accuracy is better as compared to the classic non-modular network. However, these methods still have some drawbacks. Firstly, for the methods proposed by Anand et al. in 1995 and Guan and Li in 2002, although the dimension (number of output class) of each sub-problem is smaller than the

original problem, the size of each sub-problem's training pattern set is still as large as the original problem. Therefore, each module will have unnecessarily long training time and ineffective learning especially when the original problem is very large. Secondly, the methods proposed by Anand et al. in 1995 and Lu and Ito in 1999 usually split the problem into a set of two-class sub-problems. When the original *K*-class problem is very complex (*K* is very large), a very large number of modules will be needed to learn the sub-problems and thus resulting in excessive computational cost. Thirdly, the methods proposed by Anand et al. in 1995, Lu and Ito in 1999 and Guan and Li in 2002 integrate all the modules together at the final stage in order to produce the overall solution for the original problem. This allows error from any of the modules affecting the performance (accuracy) of the other modules and thus causing interferences among the modules. During the classification process for each input pattern, all the modules have to classify that input pattern correctly. Any module classifies the input pattern wrongly may cause the overall classification process to be incorrect. Lastly, the method proposed by Lu et al. in 1994 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* to overcome the drawbacks as mentioned above. In section 3, the design details and overview of the proposed modular network architecture will be introduced. In section 4, a simple model is introduced to analysis the PD network. In section 5, modular PD is introduced to improve the performance of PD network. In section 6, the experimental results are shown and analyzed. Discussion and conclusion will be presented in section 7.

3. Design details for the pattern distributor network

In order to reduce effectively the size of the training pattern set presented to each module (small-size feedforward network) in the modular network, an additional module is incorporated into the network and it acts as a pattern distributor. This pattern distributor has a higher position (level) as compared to the other modules in the network. The overview of the new network architecture is shown in figure 3(a) whereas the specific training algorithm for the pattern distributor module is illustrated in figure 3(b) and the following section.

"Figure 3 near here"

3.1 Training of the pattern distributor network

To implement the new modular network, the first step is to decompose a complex classification problem with a large number of output classes into a set of sub-problems, each with a small number of output classes. To train the pattern distributor (module *0* as shown in figure 3(a) & (b)) that has *r* output units, first, instead of having all *N* output classes of the original training patterns presented to this module, training patterns (from *N* output classes) are first grouped together and classified into n_r classes (where $n_r < N$), namely Class n_1 patterns (patterns belonging to the original "class *1* to class *N/r*" are grouped and classified into this class), Class n_2 patterns (patterns belonging to the original "class *N/r+1* to *2N/r*" are grouped and classified into this class), …, to Class n_r patterns(patterns belonging to the original "class (*r-1)N/r+1* to class *N*" are grouped and classified into this class). This set of "manually modified training patterns" (with a smaller number of output classes) is then presented to the pattern distributor to train and help it learn.

It should be mentioned that the "equally" grouping of the output class as illustrated earlier or as shown in figure 3(a) is just to serve as a clearer example. In fact, the grouping process is flexible (based on the user's decision). Different grouping of the output classes will

cause the new modular network to have different training time and generalization accuracy. The remaining modules (module *1* to module *r* as indicated in figure 3(a)) in the network are trained by using the corresponding training patterns only (for example, training patterns belonging to "class *1* to class *N/r*" are used to train module *1*). This process continues until all the modules are well trained. Therefore, the size of training patterns presented to each module is reduced significantly as compared to the task decomposition methods mentioned in section 2.

It should be mentioned that these modules can be further decomposed (based on output parallelism) into smaller sub-modules. Thus, each module can be viewed as a group of sub-modules that are connected in parallel. The modified version of figure 3(a) is shown in figure 4:

"Figure 4 near here"

3.2 Operation of the pattern distributor network

After the training process is completed, when a new, unseen input pattern (for example pattern that belongs to "class *1* to class *N/r*") is presented to the modular network, the pattern distributor will first accept this pattern, classify it correctly and the corresponding output unit in the pattern distributor (for this example, output unit *1*) will have the largest value among all the other output units. Thus only the corresponding module (for this example, module *1*) will be activated and used. After that, the input pattern is presented to this module (module *1*) only and then this module will complete the classification process. Only two instead of all modules are used in each classification process, this is likely to reduce errors.

Constructive Backpropagation (CBP) algorithm was used to train the network in the experiments (Lehtokangas, 1999). CBP is briefly introduced in *Appendix I*. CBP can reduce

the excessive computational cost significantly and also it does not require any prior knowledge concerning decomposition.

In this paper, RPROP is used with the following parameters: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_0 = 0.1$, $\Delta_{max} = 50$, $\Delta_{min} = 1.0e\text{-}6$, with initial weights selecting from –0.25… 0.25 randomly (Riedmiller and Braun, 1993).

In order to avoid large computational cost and overfitting, a method called *early stopping* using validation set is used as the stopping criteria. The details and various definitions of the stopping criteria are presented in *Appendix II*.

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 set is 50%, 25% and 25% of the problem's total available patterns.

4. Analysis of the pattern distributor network

The performance of PD module greatly affects the performance of the whole network. When this pattern distributor classifies a pattern wrongly, the remaining classification process will also be wrong. In our design, we hope PD networks could have little or no error compared with ordinary TD networks. So the error of PD module could not be very large. We present a simple model to discuss what condition a PD network should satisfy for it to outperform an ordinary TD network.

"Figure 5 near here"

Referring to figure 5, assume the PD module has two outputs, and the number of the output classes is *K*. Both Module 1 and Module 2 have *K/2* output classes (assume K is a even number here). The network has been further divided into some sub-modules. The network is divided into the same sub-modules using an ordinary TD network (here we have chosen an output parallelism network (Guan and Li, 2000 and Guan and Li 2002)) in order to compare the results (figure 6).

"Figure 6 near here"

Consider the course of testing. In the following, we assume that each corresponding module in these two network models has the same probability of error as they are implemented in the same way. Then the error incurred from the PD network model will be the error from the pattern distributor module plus the error from the module involved, while the error from the TD network model will be the sum of errors from all the modules that respond with some incorrect results.

Assume the probability of error in the above TD network is p_e. And for each test example, the probability of error in either Module 1 or Module 2 is $p_e/2$. To those examples which could enter Module 1 of the PD network, the probability of error in Module 1 is equal to that in Part 1 of the TD network. In other words, probability of error in Module 1 is $p_e/2$. Also, the probability of error in Module 2 is $p_e/2$. Assume the probability of error in the PD module is p_{ePD}.

Assume the number of the test examples is *N*, and the number of examples belonging to Module 1 of the PD network is *N/2*.

The number of examples classified or recognized wrongly by the TD network is:

$$N_{TD} = N \cdot p_e \tag{1}$$

The number of examples dispatched wrongly by the PD network is:

$$N_{PD} = N \cdot p_{ePD} + 2 \cdot \frac{1}{2} \cdot (N - N \cdot p_{ePD}) \cdot \frac{p_e}{2} = N \cdot p_{ePD} + N \cdot \frac{p_e}{2} - N \cdot p_{ePD} \cdot \frac{p_e}{2} \tag{2}$$

If the PD network has better result than the TD network, then $N_{PD} < N_{TD}$ must be satisfied.

Notice in Equation (2), the last term is much smaller than the other two.

$$N_{PD} = N \cdot p_{ePD} + N \cdot \frac{p_e}{2} - N \cdot p_{ePD} \cdot \frac{p_e}{2} < N \cdot p_{ePD} + N \cdot \frac{p_e}{2} < N \cdot p_e = N_{TD} \tag{3}$$

So

$$p_{ePD} < \frac{1}{2} \cdot p_e \tag{4}$$

If the above relationship could be satisfied, the error of the PD network will be smaller than that of the TD network.

Discussions:

In the above analysis, *K* is considered as an even number. Here we discuss the situation that *K* is odd. Assume that PD has two outputs, and each output corresponds to a module. Module 1 has *(K+1)/2* output classes and Module 2 has *(K-1)/2* output classes. Still, assume the probability of error in the corresponding TD network is p_e. And for each test example, the probability of error in either Module 1 or Module 2 is $p_e/2$.

Assume the number of the test examples is *N*, and the number of examples belonging to each output class *N/K*. The number of examples dispatched wrongly by the PD network is:

$$\begin{aligned} N_{PD} &= N \cdot p_{ePD} + \frac{1}{K}\frac{K+1}{2} \cdot (N - N \cdot p_{ePD}) \cdot \frac{p_e}{2} + \frac{1}{K}\frac{K-1}{2} \cdot (N - N \cdot p_{ePD}) \cdot \frac{p_e}{2} \\ &= N \cdot p_{ePD} + (N - N \cdot p_{ePD}) \cdot \frac{p_e}{2} \end{aligned} \tag{5}$$

Under this assumption, if the relationship in Eq. (4) is satisfied, the PD network still could get better results.

This simple model shows if the error of PD module is small enough, the PD network could have better results than the ordinary TD network. The same analysis can be easily extened to the case when the number of modules considered is more than two.

5. Improvement on the pattern distributor network– modular pattern distributor

Results in "Parallel growing and training of neural networks using output parallelism" showed that the training time and generalization accuracy of modular networks based on output parallelism are better than classic non-modular networks (Guan and Li, 2002). Thus, instead of using a non-modular pattern distributor module (as indicated in figure 3(a)) in the network, the performance can be further improved by using a modular pattern distributor module. The output parallelism method is applied to the non-modular pattern distributor module by decomposing it into several sub-modules. An overview of the modular pattern distributor architecture is shown in figure 7:

"Figure 7 near here"

In figure 7, the pattern distributor is decomposed into 2 modules only (to simplify the figure). In fact, the number of modules is determined by the user. The performance of the modular network is expected to be better if the number of modules used is larger.

6. Experimental results and analysis

6.1 Experiment scheme

Four benchmark classification problems, namely *Vowel*, *Glass*, *Segmentation*, and *Letter Recognition* were used to evaluate the performance of the new modular network – *Task Decomposition with Pattern Distributor*. These classification problems were taken from the PROBEN1 benchmark collection (Prechelt, 1994) and University of California at Irvine (UCI) repository of machine learning database. In the set of experiments undertaken, the first three classification problems were conducted 10 trials and the *Letter Recognition* problem was conducted 5 trials (due to the long training time). All the hidden units and output units use the sigmoid activation function and E_{th} is set to 0.1. When a hidden unit needs to be added, 8 candidates are trained and the best one is selected. All the experiments were simulated on a Pentium IIII – 2.4GHZ PC. The sub-problems were solved sequentially and the CPU time expended was recorded respectively.

6.2 Experimental results and analysis

Three important metrics, namely, training time, generalization accuracy, and network complexity will be used as the criteria to judge the performance of the new modular network. Guan and Li's (2002) results showed that the performance (in terms of the three important issues as mentioned earlier) of output parallelism is better than the classic non-modular neural network. In this paper, the performance of output parallelism will be used as a yardstick and the performance of the new modular network will be compared to it.

For training time, the CPU time spent to train the modules will be compared. It should be noted that the number of training patterns presented to each module is different. Therefore, the computational cost of one training epoch can differ significantly. Comparing the number of epochs solely will lead to unfair comparison and thus training epoch will not be used as the

judging criteria. For generalization accuracy, classification error instead of test error will be compared as all the problems used are classification problems. For network complexity, the number of hidden units and independent parameters (the number of weights and biases) in the network will be compared.

A. Glass

This data set is used to classify glass types. The results of a chemical analysis of glass splinters (percentage of 8 different constituent elements) plus the refractive index are used to classify a sample to be either float processed or non-float processed building windows, vehicle windows, containers, tableware, or head lamps. This data set consists of 9 inputs, 6 outputs, and 643 patterns (they are divided into 321 training patterns, 161 validation patterns, and 161 test patterns). The patterns were normalized and scaled so that each component lies within [0, 1]. This problem was divided into two outputs by PD module. Each output of the PD module consisted of 3 outputs. Then this problem was divided into 6 sub-modules and each module has one output unit.

"Table 1 near here"

From Table 1, it is observed that the classification error using ordinary TD network (here output parallelism) was 14.2236%, and that using a non-modular PD network was 7.82609%. The classification error of a non-modular PD module was 2.422358%. According to our analysis, if Equation (4) could be satisfied, in other words, if the error of PD module was smaller than half of the error of TD network, the PD network will have better results. Equation (4) was apparently satisfied, and using the PD network had smaller classification error. It matched with our analysis. It could be also found that the classification error was further reduced when using a modular PD network compared with a non-modular network. The modular PD module's classification error was 2.36026% which was small than the non-

modular PD module. Our analysis suggests that the better performance of PD module could get better performance of the whole PD network. The overall classification error is reduced to 7.63975% when using the modular PD network.

From Table 1, it is also seen that the training time using ordinary TD network is 63.7 s in parallel and 197.7 s in series, and that using non-modular PD network is 82.9 s in parallel and 194.3 s in series. It is not have much difference. The number of hidden units and the number of independent parameters using ordinary TD network is 253.5 and 2848.5 respectively, while those using non-modular PD network is 391.2 and 4413.8 respectively. The PD network has more hidden units and independent parameters than ordinary TD network. If modular PD network is used, there are more the hidden units and independent parameters compared with non-modular PD network, but the training time in parallel is reduced.

B. Vowel

The input patterns of this data set are 10 element real vectors representing vowel sounds which 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 were normalized and scaled so that each component lies within [0, 1]. This problem was divided into three outputs by the PD module. The first output of PD consisted of 3 outputs, the second output of PD consisted of 4 outputs, and the last output of PD consisted of 4 outputs. Then this problem was divided into 11 sub-modules and each sub-module has only one output unit. Table 2 shows that the classification error using an ordinary TD network (here output parallelism) is 25.54955%, and that using a non-modular PD network is 18.70445%. The classification error of the non-modular PD module is 6.680157%. Compared to the classification error of TD network, the PD module's classification error is very small. So the PD network produced better results. It can also be found that the classification error could be

further reduced when using a modular PD network compared with a non-modular network. The overall classification error is reduced to 18.3% when using the modular PD network.

"Table 2 near here"

From Table 2, it is also seen that the training time using ordinary TD network is 58.7 s in parallel and 418.9 s in series, and that using non-modular PD network is 117 s in parallel and 245.6 s in series. Though the training time in parallel increases using PD network compared with ordinary TD network, the training time in series greatly decreases. The number of hidden units and the number of independent parameters using ordinary TD network is 184.4 and 2333.5 respectively, while those using non-modular PD network is 229.4 and 2955.8 respectively. The PD network has more hidden units and independent parameters than ordinary TD network. If modular PD network is used, there are more the hidden units and independent parameters compared with non-modular PD network, but the training time in parallel is reduced.

C. Segmentation

This data set consists of 18 inputs, 7 outputs, and a total of 2310 patterns (1155 training patterns, 578 validation patterns, and 577 test patterns). The patterns were normalized and scaled so that each component lies within [0, 1]. This segmentation problem was divided into two outputs by PD module. One of the PD's outputs consisted of 3 outputs while the other consists of 4 outputs. And the problem was divided into 7 sub-modules.

"Table 3 near here"

Table 3 shows that the classification error using an ordinary TD network (here output parallelism) is 5.181979%, and that using a non-modular PD network is 4.61005%. The classification error of the non-modular PD module is 1.03986%. According to our analysis, if Equation (4) could be satisfied, in other words, if the error of PD module was smaller than

half of the error of TD network, the PD network will have better results. Equation (4) was apparently satisfied, and using the PD network had a smaller classification error. It matched with our analysis. Compared to the classification error of TD network, the PD module's classification error is very small. So the PD network produced better results. It can also be found that the classification error could be further reduced when using a modular PD network compared with a non-modular network. The overall classification error is reduced to 4.57539% when using the modular PD network.

From Table 3, it is seen that the training time using ordinary TD network is 610.2 s in parallel and 1719.6 s in series, and that using non-modular PD network is 213.4 s in parallel and 706.9 s in series. Both the training time in parallel and in series greatly decreases using PD network compared with ordinary TD network. The number of hidden units and the number of independent parameters using ordinary TD network is 152.1 and 3175 respectively, while those using non-modular PD network is 128.9 and 2762.9 respectively. The PD network has less hidden units and independent parameters than ordinary TD network. If modular PD network is used, there are more the hidden units and independent parameters compared with non-modular PD network, but the training time in parallel is reduced.

Deleted: ¶
¶

D. 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 total of 20000 patterns (10000 training patterns, 5000 validation patterns, and 5000 test patterns). All the patterns were normalized and scaled so that each component lies within [0, 1]. The problem was divided into 14 sub-modules. 12 of them are solved by sub-modules with 2 output units while the remaining is solved by modules with 1 output unit. The PD module has 4 outputs. The first output of PD module has 4 sub-modules (7 original output classes), the second output of PD module has 4 sub-modules (7 original

output classes), the third output of PD module has 3 sub-modules (6 original output classes) and the last output of PD module has 3 sub-modules (6 original output classes).

"Table 4 near here"

Table 4 shows that the classification error using an ordinary TD network (here output parallelism) is 15.784%, and that using a non-modular PD network is 19.369%. Such a PD network did not produce better results. According to our former analysis, the network could not have better performance if the classification error of PD module is too large. Here the classification error of the non-modular PD module is 17.872%, and this value is very large compared to the classification error of TD network. It could explain why the non-modular PD network did not get better performance. We also notice that when using the modular PD network, the PD network produced lower classification error which is 15.444%. The possible reason may be that the PD module's performance is improved when using the modular PD.

7. Discussions and conclusions

7.1 Discussions

From the earlier section, it is shown that if the classification error of PD module is not very large, the performance of *Task Decomposition Pattern Distributor* method is better than the output parallelism method, which on the other hand has been shown to be better than the classic non-modular network method (that uses a single, large network to solve the problem) (Guan and Li, 2002). Therefore, the performance of a modular pattern distributor network is generally better than that of a classic non-modular network. However, the question is: How much is the improvement over a non-modular network? The performance comparison for these two networks is presented here. The *Vowel* and *Letter recognition* data sets were used in the experiments. Table 5 shows the experimental results for the three methods (*non-modular network*, *output parallelism* and *Modular Pattern Distributor* method) by using the

Vowel data set. From Table 5, it is observed that the classification error reduction by the *Task Decomposition with* ***Modular*** *Pattern Distributor* (18.3) vs. *Classic Non-modular Network* (34.737) is 47.32%. The percentage of training time reduction by the *Task Decomposition with* ***Modular*** *Pattern Distributor* (51.8) vs. *Classic Non-modular Network* (197.93) is 73.83%.

"Table 5 near here"

Table 6 shows the experimental results by using the *Letter recognition* data set*:*

"Table 6 near here"

From Table 6, it is observed that the classification error reduction by the *Task Decomposition with Modular Pattern Distributor* (15.444) vs. *Classic Non-modular Network* (21.672) is 28.74%. The percentage of training time reduction by the *Task Decomposition with Modular Pattern Distributor* (2293.6) vs. *Classic Non-modular Network* (20845.05) is 89.0%.

1. From Table 5 and 6, it is observed that the performance (in terms of training time and classification error) of a modular pattern distributor network is much better than that of a classic non-modular network. The reduction in training time is especially significant (>73%).Lastly, all the experimental results showed that the new modular network inherits the advantages provided by the output parallelism method.

Besides these advantages, the new modular network provides more advantages (as compared to the output parallelism method):

1. Training time is further reduced since each module in the network only need to solve a smaller and simpler sub-problem. The reduction is significant when the size of the original training pattern set is large.
2. Generalization accuracy is further improved since all the modules in the new modular network can solve the sub-problems better.

3. Although the total number of independent parameters in the new modular network exceeds that in the output parallelism method, the new modular approach yields faster convergence.
4. Various combinations of modules (in parallel and in series) allow more useful and flexible problem solving as compared to the output parallelism method, which only uses parallel combination.

In order to further improve the PD method, we could also apply the *Pattern Distributor* method to the PD module. In other words, multi-level pattern distributors (performing task decomposition by applying the *pattern distributor* method to the pattern distributor module) could be considered.

7.2 Conclusions

This paper presented a better (as compared to the output parallelism approach or conventional non-modular approach) task decomposition approach called *Task Decomposition with Pattern Distributor* to build a new modular network. This new approach not only inherits the advantages provided by the output parallelism method but also provides more advantages. Its performance can be improved further by incorporating additional pattern distributor modules into the network. Based on this method, a problem can be divided flexibly into several sub-problems by the pattern distributor module, where each sub-problem is composed of the whole input vector and a fraction of the output vector. The combinations (in parallel and in series) of modules in the new modular network were used to solve each sub-problem respectively. This new method could not only reduce the internal interferences that exist inside the hidden structure of the large network by decoupling it into several modules but also prevent the error from any of the modules affecting the performance (accuracy) of the other

modules by designing all the modules independent from each other. Besides, this new method also builds modules that can solve the sub-problems better and faster since by incorporating the pattern distributor module into the network, the size (number of patterns) and dimension (number of output classes) of training pattern set presented to each sub-module would be reduced, thus, unnecessary long training time and ineffective learning can be avoided

Our analysis and the experimental results showed that this new method has shorter training time and better generalization accuracy as compared to the output parallelism method. The results of this new method could be further improved by using more levels of modules in the network.

References

Anand, R., Mehrotra, K., Mohan, C. K. and Ranka, S. (1995) Efficient classification for multiclass problems using modular neural networks, *IEEE Transactions on Neural Networks*, 6(1), 117 – 124.

Baum, E. B. and Haussler, D. (1989) What size net gives valid generalization, *Neural Computation*, 1(1), 151-160.

Blum, A. and Rivest, R. L. (1992) Training a 3-node neural network is NP-complete, *Neural Networks*, 5(1), 117-128.

Guan, S. U. and Li, S. C. (2000) An approach to parallel growing and training of neural networks, *Proceeding of 2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, Honolulu, Hawaii, 2, 1101 – 1104.

Guan, S. U. and Li, S. C. (2002) Parallel growing and training of neural networks using output parallelism, *IEEE Transactions on Neural Networks*, 13(3), 542 -550.

Jacobs, R. A., Jordan, M. I., Nowlan, M. I. and Hinton, G. E. (1991) Adaptive mixtures of local experts, *Neural Computation*, 3(1), 79-87.

Lehtokangas, M. (1999) Modeling with constructive backpropagation, *Neural Networks*, 12, 707-716.

Lu, B. L. Kita, H., and Nishikawa, Y. (1994) A multisieving neural-network architecture that decomposes learning tasks automatically, *Proceedings of IEEE Conference on Neural Networks*, Orlando, FL, 1319-1324.

Lu, B. L. and Ito, M. (1999) Task decomposition and module combination based on class relations: A modular neural network for pattern classification, *IEEE Transactions on Neural networks*, 10(5), 1244 – 1256.

Prechelt, L. (1994) PROBEN1: A set of neural network benchmark problems and benchmarking rules, *Technical Report 21/94*, Department of Informatics, University of Karlsruhe, Germany.

Prechelt, L. (1997) Investigation of the CasCor family of learning algorithms, *Neural Networks*, 10(5), 885 – 896.

Riedmiller, M. and Braun, H. (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, 586-591.

Appendix I

The Constructive Backpropagation algorithm (CBP) can be depicted briefly as follows (Lehtokangas, 1999 and Guan and Li, 2002):

1. *Initialization*: The network has no hidden units. Only bias weights and shortcut connections from the input units to the output units feed the output units. Train the weights of this initial configuration by minimizing the sum of squared errors:

$$E = \sum_{p=1}^{P} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2 \qquad (1)$$

where P is the number of training patterns, K is the number of output units, o_{pk} is the actual output value of the k th output unit for the p th training pattern and t_{pk} is the desired output value of the k th output unit for the p th training pattern.

"Figure 8 near here"

2. *Training a new hidden unit*: Connect inputs to the new unit (let the new unit be the i th hidden unit, $i > 0$) and connect its output to the output units as shown in Figure 8. Adjust all the weights connected to the new unit (both input and output connections) by minimizing the modified sum of squared errors:

$$E_i = \sum_{p=1}^{P} \sum_{k=1}^{K} \left(a(\sum_{j=0}^{i-1} w_{jk} o_{pj} + w_{ik} o_{pi}) - t_{pk} \right)^2 \qquad (2)$$

where w_{jk} is the connection from the j th hidden unit to the k th output unit (w_{0k} represents a set of weights which are the bias weights and shortcut connections trained in step 1), o_{pj} is the output of the j th hidden unit for the p th training pattern (o_{p0} represent inputs to bias weights and shortcut connections), and $a(\cdot)$ is the activation function. Note that in the new

ith unit perspective, the previous units are fixed. In other words, we are only training the weights connected to the new unit (both input and output connections).

3. *Freezing a new hidden unit*: Fix the weights connected to the unit permanently.

4. *Testing for convergence*: If the current number of hidden units yields an acceptable solution, then stop the training. Otherwise go back to step 2.

Appendix II

The *Early Stopping* method using validation set is used as the stopping criteria in training the new modular network. 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 set is 50%, 25% and 25% of the problem's total available patterns. The error measure E used is *the squared error percentage* (Prechelt, 1994), derived from the normalization of the mean squared error to reduce the dependency on the number of coefficients in the problem representation and on the range of output values used:

$$E = 100 \cdot \frac{o_{\max} - o_{\min}}{K \cdot P} \sum_{p=1}^{P} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2 \qquad (3)$$

where $o_{\max}$ and $o_{\min}$ are the maximum and minimum values of output coefficients in the problem representation.

$E_{tr}(t)$ is the average error per pattern of the network over the training set, measured after epoch t. The value $E_{va}(t)$ is the corresponding error on the validation set after epoch t and is used by the stopping criterion. $E_{te}(t)$ is the corresponding error on the test set; it is not known to the training algorithm but characterizes the quality of the network resulting from training.

The value $E_{opt}(t)$ is defined to be the lowest validation set error obtained in epochs up to epoch t:

$$E_{opt}(t) = \min_{t' \le t} E_{va}(t') \qquad (4)$$

The *generalization loss* (Prechelt, 1994) at epoch t is defined as the relative increase of the validation error over the minimum so far (in percent):

$$GL(t) = 100 \cdot (\frac{E_{va}(t)}{E_{opt}(t)} - 1) \qquad (5)$$

A high generalization loss is one candidate reason to stop training because it directly indicates overfitting.

To formalize the notion of training progress, a *training strip of length* m (Prechelt, 1994) is defined to be a sequence of m epochs numbered $n+1 \ldots n+m$ where n is divisible by m. The training progress measured after a training strip is:

$$P_m(t) = 1000 \cdot (\frac{\sum_{t' \in t-m+1 \ldots t} E_{tr}(t')}{m \cdot \min_{t' \in t-m+1 \ldots t} E_{tr}(t')} - 1) \qquad (6)$$

It is used to measure how much larger the average training error is than the minimum training error during the training strip.

During the process of growing and training individual modules, we adopted the following heuristic overall stopping criteria: $E_{opt} < E_{th}$ **OR** *(Reduction of training set error due to the last new hidden unit is less than 0.01%* **AND** *Validation set error increased due to the last new hidden unit)*. The first part ($E_{opt} < E_{th}$) means that the optimal validation set error is below the threshold (E_{th}) and the result has been acceptable. The other part means the last insertion of a hidden unit resulted in hardly any progress. The criteria for adding a new hidden unit are as follows: *At least 25 epochs reached for the current network* **AND** *(Generalization loss* $GL(t) > 5$ ***OR*** *Training progress* $P_5(t) < 0.1$*)*. The first part means that the current network should be trained for at least a certain number of epochs before a new hidden unit is installed because the error curves may be turbulent at the beginning. The second part means that the current network has been overfitted or training has little progress. It is a bit unsatisfactory that all of these criteria are heuristic.

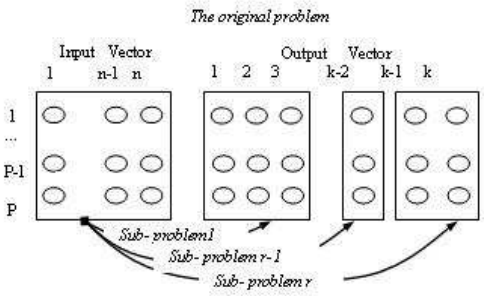

Figure 1: Problem decomposition based on Output Parallelism

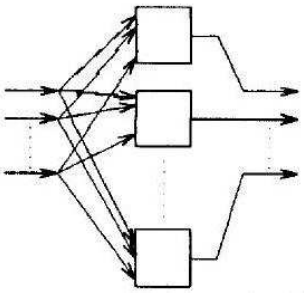

Figure 2: Task Decomposition based on Output Parallelism

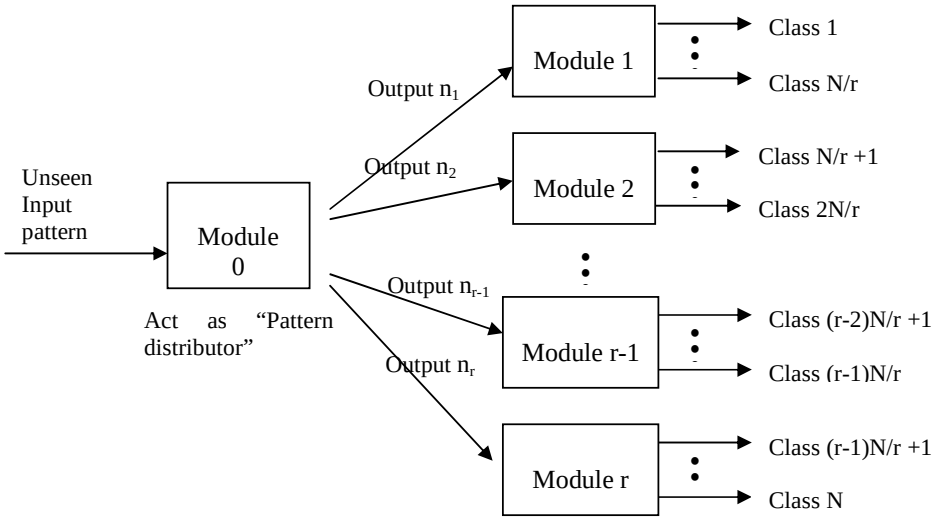

Figure 3(a): Overview of the pattern distributor network architecture

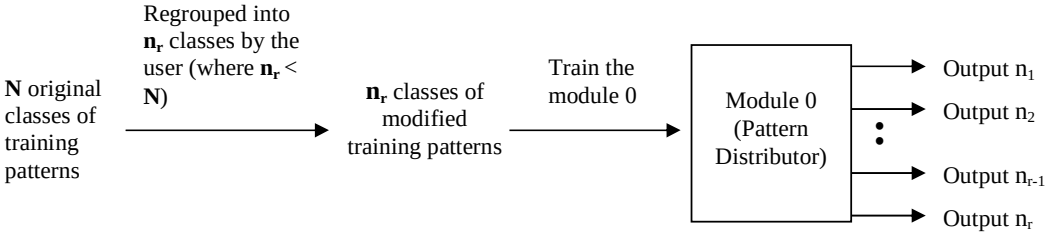

Figure 3(b): Training for the pattern distributor module

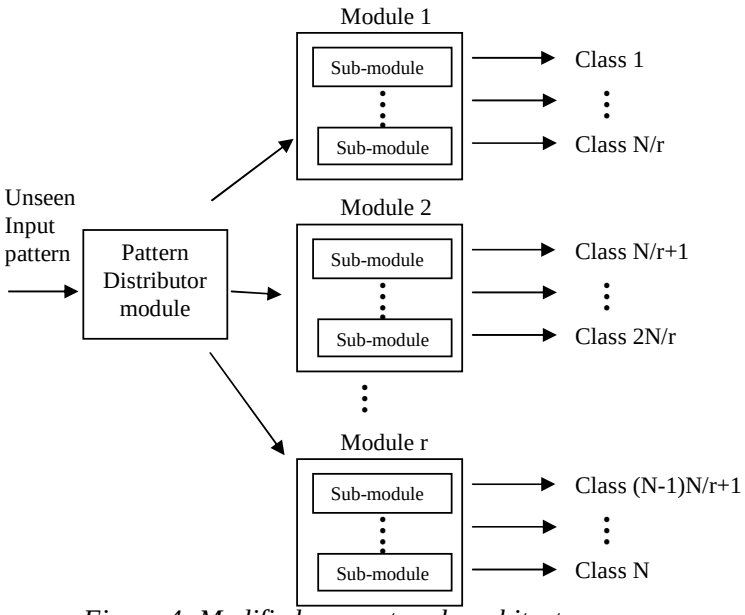

Figure 4: Modified new network architecture

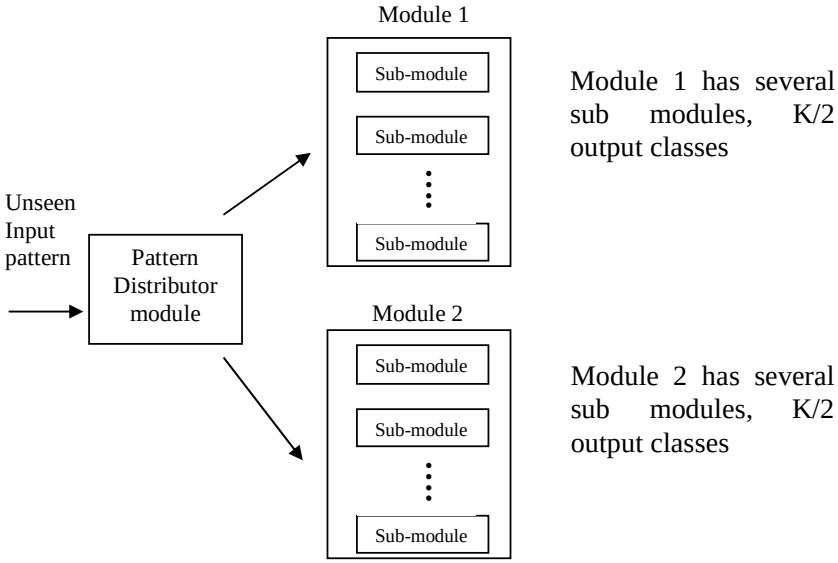

Figure 5: Pattern Distributor divides the output classes into two equal portions.

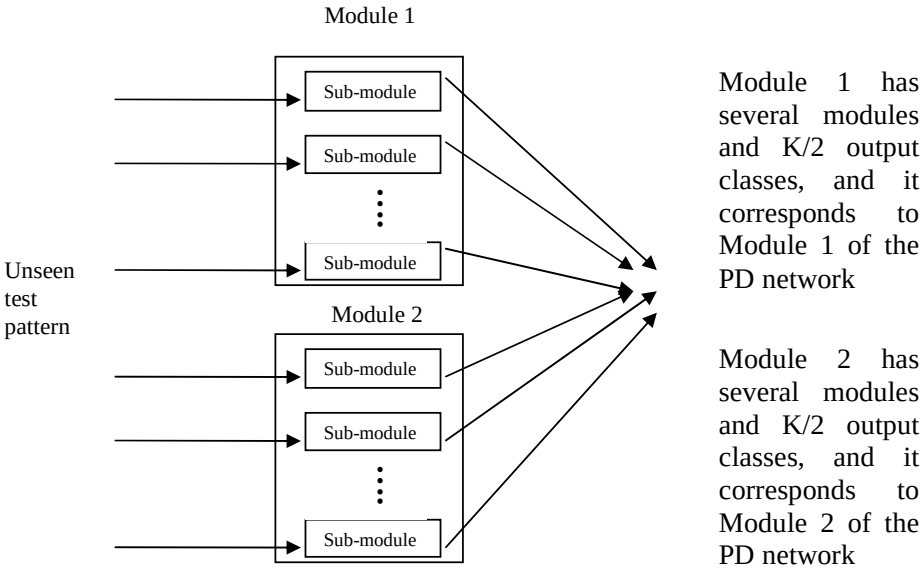

Figure 6: An ordinary TD network corresponding to the PD network in figure 5.

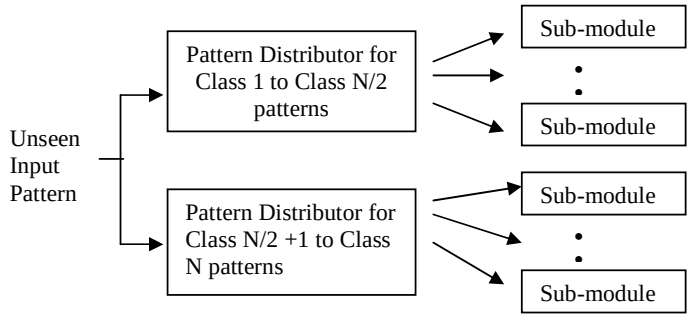

Figure 7: Modular pattern distributor

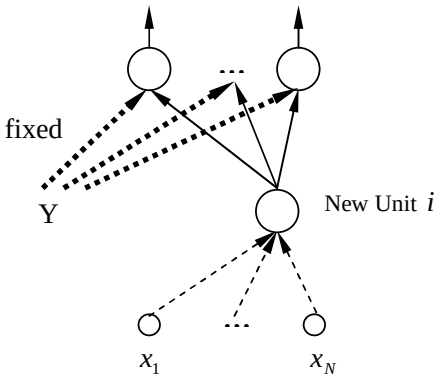

Figure 8: Training a new hidden unit in CBP learning. Y represents previously added connections to network output units.

Task Decomposition Method	Training time (s)	Hidden Units	Indp. Param.	C. error (%)
Output Parallelism (6 sub-modules)	63.7 (in parallel) 197.7 (in series)	253.5	2848.5	14.2236
Non-modular Pattern Distributor's performance (each output of PD module include three sub-modules)	82.9	30.6	387.2	2.422358
Task Decomposition with **non-modular** pattern distributor (1 pattern distributor modules and 6 sub-modules)	82.9 (in parallel) 194.3 (in series)	391.2	4413.8	7.82609
Modular Pattern Distributor's Overall Performance	54.7 (in parallel) 108.7 (in series)	59.7	676.7	2.360246
Task Decomposition with **modular** pattern distributor (2 pattern distributor modules and 6 sub-modules)	54.7 (in parallel) 220.1 (in series)	420.3	4703.3	7.63975

Table 1: Results for the Glass data

NOTES: 1. In the "Task Decomposition Method" column, "**non-modular** pattern distributor" means the pattern distributor module is a classic non-modular feedforward network while "**modular** pattern distributor" means the pattern distributor module is decomposed into several modules based on the *Output parallelism* method.

2. "Training time" column stands for the time (CPU time, in seconds) taken by growing and training each module. Training time (in parallel) stands for the maximum training time among all the modules (all modules are trained in parallel). Training time (in series) stands for the sum of training time for all the modules (all modules are trained in series).
3. "Indp. Param." stands for the total number of independent parameters (the number of weights and biases in the network) of all modules.
4. "C. Error" stands for classification error.

Task Decomposition Method	Training time (s)	Hidden Units	Indp. Param.	C.error (%)
Output Parallelism	58.7	184.4	2333.8	25.54655

(11 modules)	(in parallel) 418.9 (in series)			
Non-modular Pattern Distributor's performance (outputs of PD module include 3, 4, 4 output sub-modules respectively)	117	24.5	376	6.680157
Task Decomposition with **non-modular** pattern distributor (1 pattern distributor modules and 11 sub-modules)	117 (in parallel) 245.6 (in series)	229.4	2955.8	18.70445
Modular Pattern Distributor's Overall Performance	51.8 (in parallel) 138.8 (in series)	54	681	6.072874
Task Decomposition with **modular** pattern distributor (3 pattern distributor modules and 11 sub-modules)	51.8 (in parallel) 267.4 (in series)	258.9	3260.8	18.3

Table 2: Results for the Vowel data

NOTES: Refer to NOTES under Table 1.

Task Decomposition Method	Training time (s)	Hidden Units	Indp. Param.	C.error (%)
Output Parallelism (7 modules)	610.2 (in parallel) 1719.6 (in series)	152.1	3175	5.181979
Non-modular Pattern Distributor's performance (outputs of PD module include 3 and 4 output sub-modules respectively)	213.4	13.9	329.9	1.03986
Task Decomposition with **non-modular** pattern distributor (1 pattern distributor modules and 7 sub-modules)	213.4 (in parallel) 706.9 (in series)	128.9	2762.9	4.61005
Modular Pattern Distributor's Overall Performance	155.6 (in parallel) 302.6 (in series)	24.3	524	1.091853
Task Decomposition with **modular** pattern distributor (2 pattern distributor modules and 7 sub-modules)	155.6 (in parallel) 796.1 (in series)	139.3	2957	4.57539

Table 3: Results for the Segmentation data

NOTES: Refer to NOTES under Table 1.

Task Decomposition Method	Training time (s)	Hidden Units	Indp. Param.	C.error (%)

Output Parallelism (14 modules)	3707 (in parallel) 29483.2 (in series)	394.6	7877	15.784
Non-modular Pattern Distributor's performance (outputs of PD module include 4,4, 3 and 3 output sub-modules respectively)	3940.8	73	1601	17.872
Task Decomposition with **non-modular** pattern distributor (1 pattern distributor modules and 14 sub-modules)	3940.8 (in parallel) 15088.8 (in series)	460.4	9331.8	19.396
Modular Pattern Distributor's Overall Performance	2293.6 (in parallel) 7610.4 (in series)	194.6	3570.8	13.088
Task Decomposition with **modular** pattern distributor (4 pattern distributor modules and 14 sub-modules)	2293.6 (in parallel) 18758.4 (in series)	582	11301.6	15.444

Table 4: Results for the Letter Recognition data

Method	Training time (s)	Hidden Units	Indp. Param.	C.error (%)
Classic Non-modular Network	197.93	26.65	707	34.737
Output Parallelism (11 modules)	58.7 (in parallel) 418.9 (in series)	184.4	2333.8	25.54655
Task Decomposition with modular pattern distributor (2 pattern distributor modules and 7 sub-modules)	51.8 (in parallel) 267.4 (in series)	258.9	3260.8	18.3

Table 5: Results for the Vowel data

Method	Training time (s)	Hidden Units	Indp. Param.	C.error (%)
Classic Non-modular Network	20845.05	73.6	3607	21.672
Output Parallelism (14 modules)	3707 (in parallel) 29483.2 (in series)	394.6	7877	15.784
Task Decomposition with modular pattern distributor (4 pattern distributor modules and 14 sub-modules)	2293.6 (in parallel) 18758.4 (in series)	582	11301.6	15.444

Table 6: Results for the Letter Recognition data