Reduced Pattern Training in Pattern Distributor Networks

Chunyu Bao and TseNgee Neo

Department of Electrical and Computer Engineering
National University of Singapore

Sheng-Uei Guan

School of Engineering and Design
Brunel University, UK

In this paper, we propose a new task decomposition method, Task Decomposition with Pattern Distributor *(PD), for multilayered feedforward neural networks. The method uses a combination of network modules in parallel and series to generate the overall solution for a complex problem. We also introduce a method called* reduced pattern training *in PD networks. This method aims to improve the performance of the pattern distributor network. Our analysis and the experimental results show that reduced pattern training improves the performance of pattern distributor network significantly. Experimental results confirm that this new method can reduce training time and improve network generalization accuracy significantly when compared to ordinary task decomposition methods such as* Output Parallelism.

ACM Classification: I.2

Keywords – Task decomposition, pattern distributor, reduced pattern training, full pattern training

1. INTRODUCTION

Multilayered feedforward neural networks 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 (Jacobs *et al,* 1991). The influences 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 during the weight-updating process (Guan and Li, 2002). In order to overcome this drawback, various task decomposition methods have been proposed (Anand *et al,* 1995; Lu and Ito, 1999; Guan and Li, 2000; Guan and Li, 2002; Guan and Liu, 2004; Guan and Zhu, 2004; Lu *et al*, 1994). Instead of using a single, large feedforward network (classic network), task decomposition methods divide a problem into a set of smaller and simpler sub-problems based on "divide-and-conquer". The results obtained from solving these sub-problems are integrated together and the solution for the original problem is obtained.

Anand *et al* (1995) proposed a method that splits a *K*-class problem into *K* two-class sub-problems. Each sub-network is trained to learn one sub-problem only. Therefore, each sub-network 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 proposed by Lu and Ito

Manuscript received: 7 June 2006
Communicating Editor: John Yearwood

(1999) divides a *K*-class problem into $\binom{K}{2}$ two-class sub-problems. A module is designated to learn each sub problem 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 above class decomposition method, *Output Parallelism*, is proposed by Guan (Guan and Li, 2000; Guan and Li, 2002; Guan and Liu, 2004; Guan and Zhu, 2004). Using output parallelism, a complex problem can be divided into several sub-problems as chosen, each of which is composed of the whole input vector and a fraction of the output vector. Each module (for one sub-problem) 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 a problem with a high dimensional output space into several sub-problems with lower dimensional output spaces, the method proposed by Lu *et al* (1994) decomposes the problem into several sub-problems in smaller sizes. 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 the progress of training.

Although these methods are efficient, there are still some drawbacks associated with them. Firstly, the methods proposed by Anand (1995) and Lu and Ito (1999) usually split the problem into a set of two-class sub-problems. If 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. Secondly, for the methods proposed by Anand (1995) and Guan and Li (2002), although the dimension (number of output classes) of each sub-problem is smaller than

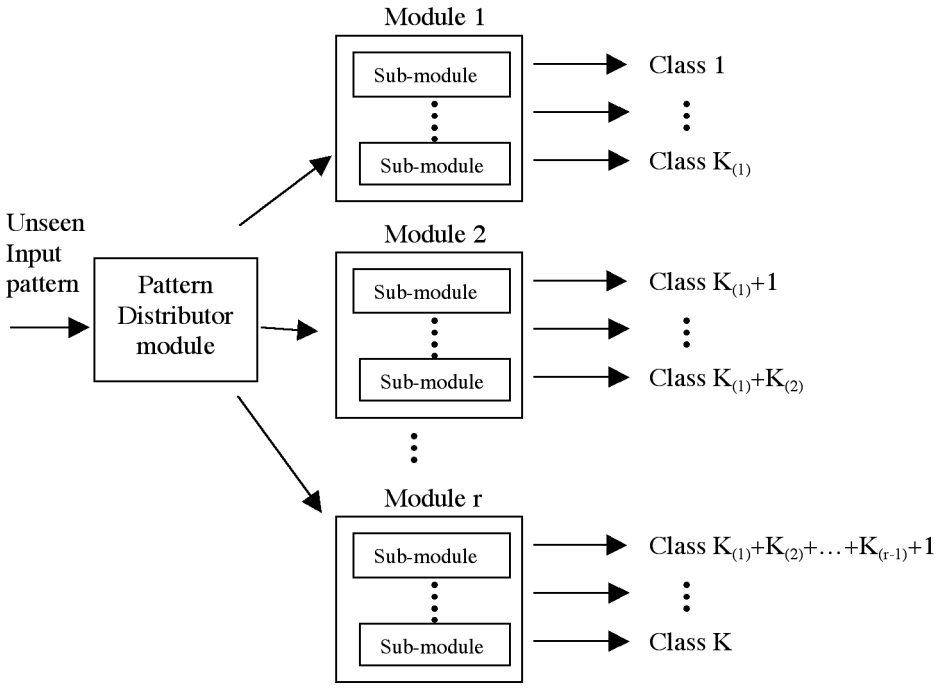


Figure 1: A typical PD network

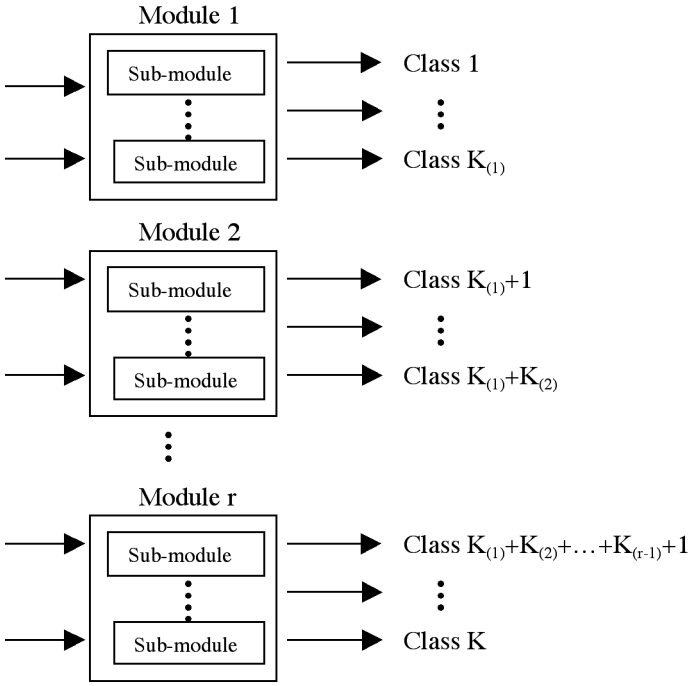


Figure 2: An Output Parallelism network

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 with many training patterns. Lastly, the method proposed by Lu *et al* (1994) only reduces the size of the problem but not the dimension of the problem. The internal interferences (that exist 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 above. A special module called a Pattern Distributor (PD) module is introduced to improve the performance of the whole network. This PD module stands at a higher position as compared to the other modules in the network, which means an unseen input pattern will be processed by the PD module first. A typical PD network is shown in Figure 1.

In this paper, the PD method will be discussed in detail. In Section 2, the operation of PD networks is introduced. In Section 3, we present a method called reduced pattern training for PD networks. This method improves significantly the performance of PD networks. In Section 4, the experimental results are shown and analyzed. Conclusions are drawn in Section 5.

2. FROM OUTPUT PARALLELISM TO PATTERN DISTRIBUTOR NETWORKS

The PD method is originated from Output Parallelism (OP). Figure 2 is a typical OP network. OP can be regarded as a typical task decomposition networks. In an OP network, each module can be learned separately using all the training patterns.

In order to improve the performance of the network, we introduce a special module called a Pattern Distributor (PD) module before other modules. Every output of the PD module has a fraction of the overall output classes of the original problem. For Module 1 to Module *r*, the ordinary task decomposition methods (here *Output Parallelism*) could be applied, so Module 1 to Module *r* could be divided into sub-modules. The PD method could shorten the training time and improve the generalization accuracy of a network compared with ordinary task decomposition methods.

In the OP network, we can train the modules (or sub-modules) separately. In the PD network, we can also train non-PD modules (or sub-modules) separately. If we use all the training patterns to train these modules, then Module 1 to Module *r* in the PD network (see Figure 1) will be the same as their counterparts in the OP network (see Figure 2). Therefore, the weights and the hidden units of the non-PD modules will be the same as those of the corresponding modules in the OP network.

After the training process of the PD network is completed and the network has been set up, when a new, unseen input pattern is presented to the PD network, the PD module will be the first to classify it. The corresponding output unit in the pattern distributor will have the largest output value among all the output units. Thus 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 PD module and the corresponding module are used in the classification process.

3. MOTIVATION FOR REDUCED PATTERN TRAINING

In a PD network, an unseen pattern is firstly classified by the PD 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 notice that each non-PD module only processes a subset of the test patterns. In other words, each non-PD module only disposes those patterns that belong to one of the classes in this module if the PD module classifies all the test patterns correctly. Also, if the PD module classifies some patterns wrongly, the mistake can not be corrected by the later module. Motivated by the function of those non-PD modules, we may consider training those modules only using the patterns belonging to the classes in the modules. The method using the corresponding training patterns to train Module *j* ($j = 1,2, \ldots r$) is called reduced pattern training. Similarly, the method of using the whole training set to train Module j ($j = 1,2, \ldots r$) is called full pattern training.

When we train Module *j* using full pattern training, the module will carry the information of the instances that do not belong to its own classes. That information does not contribute to the distribution of patterns in Module *j* to the corresponding classes. So it is useless. Also, compared with full pattern training, training time could be reduced when training is done with reduced patterns.

Moreover, training Module *j* together with unnecessary patterns may decrease the ability of Module *j* to classify the patterns belonging to Module *j* correctly. There are two aspects. Firstly, the objective of training is to let the module reach its best classification accuracy when processing the patterns dispatched to it. Using full pattern training, the 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. Secondly, we know that the patterns not belonging to Module *j* would have the outputs as 0 during the learning process of the module. (In our experiments, if a pattern belongs to some class, the corresponding output would be 1, otherwise, 0). With the introduction of those patterns not belonging to Module *j*, there are much more patterns with an output label 0 than the patterns with an output label 1 in the learning of

specific outputs. So the patterns with an output label *0* will have more influence in updating the weights. The patterns with an output label *0* will be influential in the computation of the training error function. And the patterns with an output label 1 will become less influential in the decision of weight update. 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 above observations, we can see those unnecessary patterns are harmful to the module training.

Reduced pattern training might not be applicable in 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 it may lead to poor accuracy when the test patterns are presented. How to reduce patterns in an OP network is also a problem. While in the test course of a PD network, each module (not including the PD module) only needs to deal with the test patterns belonging to this module. Reduced pattern training would not miss any useful information in the PD network. Our experimental results showed that reduced pattern training is crucial for a PD network to gain good performance.

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experimental Scheme

Constructive Backpropagation (CBP) algorithm (Lehtokangas, 1999) was used to train the network in the experiments. CBP is briefly introduced in Appendix I. CBP can reduce the excessive computational cost significantly and 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 selected from $-0.25 \ldots 0.25$ randomly (Riedmiller and Braun, 1993). In order to avoid large computational cost and overfitting, a method called early stopping based on 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 sets is 50%, 25% and 25% of the problem's total available patterns.

Four benchmark classification problems, namely *Vowel*, *Glass*, *Segmentation*, and *Letter Recognition* are used to evaluate the performance of the new modular network – *Task Decomposition with Pattern Distributor*. These classification problems are 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 times and the *Letter Recognition* problem was conducted five times (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 addition was required, eight candidates were trained and the best one selected. All the experiments were simulated on a Pentium IV – 2.4GHZ PC. The sub-problems were solved sequentially and the CPU time expended was recorded respectively.

4.2 Experiments for PD network based on full and reduced pattern training

A. 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). The patterns were normalized and scaled so that each component lies within [0, 1]. The OP network is composed of two modules, Module 1 recognizes class {1,3,5} and Module 2 recognizes class

{2,4,6}. These two modules are further decomposed into six sub-modules and each sub-module recognizes one class from all the patterns. In the PD network, the PD module has two outputs, and each output is connected to one non-PD module. Module 1 recognizes Class {1, 3, 5} while Module 2 recognizes Class {2, 4, 6}. Similar to the OP network, the two non-PD modules are further divided into six sub-modules. The sub-modules of Module 1 recognize class 1, 3, 5 respectively and the sub-modules of Module 2 recognize class 2, 4, 6 respectively.

From Table 1, we can see that the classification error obtained by the PD network with full pattern training (10%) is much smaller than that obtained by the ordinary method (16.0870%) and the OP network (14.2547%). It could be noticed that classification error is significantly decreased using reduced pattern training (7.8261%) compared to full pattern training (10.0%). The training time is also greatly reduced using reduced pattern training (194.3s in series) in contrast to full pattern training (298.7s in series), and it is even shorter than that of *output parallelism* (197.7s in series).

Method		Training time (s)	Hidden Units	Indp. Param.	C. error (%)
Ordinary method (no task decomposition)		168.1	46	796	16.0870
Output Parallelism (2 modules, 6 sub-modules)		63.7 (in parallel) 197.7 (in series)	253.5	2848.5	14.2547
Pattern Distributor (1 PD module, 2 modules and 6 sub-modules)	PD module (PD module has 2 outputs)	82.9	30.6	387.2	2.4224
	The overall network (full pattern training)	85.2 (in parallel) 298.7 (in series)	280.9	3200.5	10.0
	The overall network (reduced pattern training)	82.9 (in parallel) 194.3 (in series)	391.2	4413.8	7.8261

NOTE: 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" 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.
5. "Ordinary method" referd to the neural networks in which task decomposition is not applied. It can be also called non-modular network.

Table 1: Results for the Glass data

Method		Training time (s)	Hidden Units	Indp. 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 (in parallel) 418.9 (in series)	184.4	2333.8	25.5466
Pattern Distributor (1 PD module, 3 modules and 11 sub-modules)	PD module (PD module has 3 outputs)	117	24.5	376	6.68016
	The overall network (full pattern training)	102.7 (in parallel) 534.3 (in series)	210.6	2730.2	24.8987
	The overall network (reduced pattern training)	117 (in parallel) 245.6 (in series)	229.4	2955.8	18.7045

Table 2: Results for the vowel data

B. Vowel

The input patterns of this data set are 10 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 were normalized and scaled so that each component lies within [0, 1]. The OP network is composed of three modules. Module 1 recognizes class {1,2,3}, Module 2 recognizes class {4,5,6,7} and Module 3 recognizes class {8,9,10,11}. These three modules are further decomposed into 11 sub-modules and each sub-module recognizes one class from all the patterns. In the PD network, the PD module has three outputs and each output is connected to one non-PD module. Module 1 recognizes class {1,2,3}, Module 2 recognizes class {4,5,6,7} and Module 3 recognizes class {8,9,10,11}. Similar to the OP network, the three non-PD modules are further divided into 11 sub-modules.

From Table 2, we can see that the classification error obtained by the PD network with full pattern training (24.8987%) is smaller than that obtained by the ordinary method (37.1660%) and the OP network (25.5466%). It is noticed that classification error is significantly decreased using reduced pattern training (18.7045%) compared to full pattern training (24.8987%). The training time is greatly reduced using reduced pattern training (245.6s in series) in contrast to full pattern training (534.3s in series), and it is shorter than that of *output parallelism* (418.9s in series).

C. Segmentation

This data set consists of 18 inputs, seven 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]. The OP network is composed of two modules. Module 1 recognizes class {3,4,5} and Module 2 recognizes class {1,2,6,7}. These two modules are further decomposed into seven sub-modules and each sub-module recognizes one class from all the

Method		Training time (s)	Hidden Units	Indp. Param.	C. error (%)
Ordinary method (no task decomposition)		693.8	29	887	5.73657
Output Parallelism (7 sub-modules)		610.2 (in parallel) 1719.6 (in series)	152.1	3175	5.18198
Pattern Distributor (1 PD module, 2 modules and 7 sub-modules)	PD module (PD module has 2 outputs)	213.4	13.9	329.9	1.03986
	The overall network (full pattern training)	1002.2 (in parallel) 2219.2 (in series)	128.5	2754.9	5.44194
	The overall network (reduced pattern training)	213.4 (in parallel) 706.9 (in series)	128.9	2762.9	4.61005

Table 3: Results for the segmentation data

patterns. In the PD network, the PD module has two outputs and each output is connected to one non-PD module. Similar to the OP network, Module 1 recognizes class {3,4,5}and Module 2 recognizes class {1,2,6,7}. The two non-PD modules are further divided into seven sub-modules.

From Table 3, it is observed that the classification error using an *OP* network is 5.18198%, compared to that of the ordinary method 5.73657%. While using the PD network with full pattern training has classification error as 5.44194% and the PD network with reduced pattern training has 4.61005%. It is noticed that the classification error is decreased using reduced pattern training compared to full pattern training. Using reduced pattern training, the PD network performs better than OP and the network without task decomposition. The training time is also reduced using reduced pattern training (706.9s in series) in contrast to full pattern training (2219.2s in series), and it is also shorter than that of *output parallelism* (1719.6s in series).

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 within [0, 1]. The OP network is composed of four modules. Module 1 recognizes classes 1 - 7, Module 2 recognizes classes 8 - 14, Module 3 recognizes classes 15 - 20 and Module 4 recognizes classes 21 - 25. In the PD network, the PD module has four outputs and each output is connected to one non-PD module. Similar to the OP network, Module 1 recognizes classes 1 - 7, Module 2 recognizes classes 8 - 14, Module 3 recognizes classes 15 - 20 and Module 4 recognizes classes 21 - 25.

Table 4 shows that the classification error using an *OP* network is 19.260%, compared to that of the ordinary method 21.672%. While using the PD network with full pattern training has

Method		Training time (s)	Hidden Units	Indp. Param.	C. error (%)
Ordinary method (no task decomposition)		20845.05	73.6	3607	21.672
Output Parallelism (4 modules)		5519 (in parallel) 18112.6 (in series)	173.4	6586.8	19.260
Pattern Distributor (1 PD module, 4 modules)	PD module (PD module has 2 outputs)	2510 (in parallel) 8497 (in series)	219.5	4019.0	12.195
	The overall network (full pattern training)	6110 (in parallel) 26723.8 (in series)	386.25	8384.5	20.515
	The overall network (reduced pattern training)	2510 (in parallel) 14094.5 (in series)	344.25	7391.0	15.855

Table 4: Results for the letter data

classification error as 20.515% and the PD network with reduced pattern training has 15.855%. It is noticed that the classification error is decreased using reduced pattern training compared to full pattern training. Using reduced pattern training, the PD network performs better than OP and the network without task decomposition. The training time is also reduced using reduced pattern training (14094.5s in series) in contrast to full pattern training (26723.8s in series), and it is even shorter than that of *output parallelism* (18112.6s in series).

Discussions:

According to our experimental results, grouping classes arbitrarily in the PD module does not always lead to good classification rate. How to group the classes and combine them becomes an important issue in designing a PD network. Here we present a method to find good grouping for the PD module. The basic idea is to find classes which are close in the feature space and combine them together so that the task of PD is made easier and less error prone. Firstly, we project a *d*-dimensional (*d* is the number of the input classes) input space to a one-dimensional space using Fisher's linear discriminant (FLD) method (Duda and Hart, 1973). After that, the distances between the centres of different classes are calculated. Then these distances are arranged to form a table which is called Cross-talk table. If the distance between two classes in the Cross-talk 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 Cross-talk table. We illustrate this method below with an example.

The Cross-talk table for the Glass data set is shown in Table 5. We can see that the distances between class 1, class 2 and class 3 are 0.4467, 0.3481 and 0.501 respectively, which are much smaller than the other distances. So classes 1, 2, 3 are combined. In the remaining classes, it appears that class 4 is close to class 2. However, the distances between (4,1) and (4,3) are very large. Class

	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 5: Cross-talk table for the Glass data

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

Table 6: Results for the Glass problem using Cross-talk based combination

4 is not added to combination {1,2,3}. Note that classes 4, 5 and 6 have relatively small distances. Thus, classes 4,5,6 are combined. The final grouping is therefore {{1,2,3}, {4,5,6}}.

We use another grouping {{3,4,6},{1,2,5}}for comparison with the above set. In this set, we combined together the classes with relatively large distances. The experimental results for the two different partitions are shown in Table 6.

From Table 6, 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.

5. CONCLUSIONS

This paper presented a unique task decomposition approach called *Task Decomposition with Pattern Distributor* (PD) to build a new modular network as compared to Output Parallelism or conventional non-modular approaches. A PD network not only inherits the advantages provided by the Output Parallelism method but also provides more advantages and improves performance further by incorporating an additional PD module into the network. In order to get better performance, the method of reduced pattern training was introduced. Compared to the method of full pattern training, this method significantly reduced the classification error and the training time of a PD network. The experiments showed that these methods could improve the results.

It should be mentioned that our PD method was designed for multi-class classification problems. For problems with a large number of output classes, we suggest a multi-level PD or hierarchical PD network approach to partition the output space in a finer scale. How to extend our PD method to regression problems is also on our research agenda for future work.

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: 117–124.

DUDA, R. O. and HART, P.E. (1973): *Pattern Classification and Scene Analysis*, New York: Academic Express.

GUAN, S-U. and RAMANATHAN, K. (2004): Recursive percentage based hybrid pattern (RPHP) Training for curve fitting, In *Proc of the IEEE Int. Conf. on Cybernetics and Intelligent Systems* (CIS), 445–450,

GUAN, S-U. and LI, S. (2000): An approach to parallel growing and training of neural networks, in *Proceeding of 2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, 2: 1101–1104.

GUAN, S-U. and LI, S. (2002): Parallel growing and training of neural networks using output parallelism, *IEEE Transactions on Neural Networks*, 13: 542–550.

GUAN, S-U. and LI, P. (2002): Feature selection for modular neural network Classifiers, *Journal of Intelligent Systems*, 12: 173–200.

GUAN, S-U. and LIU, J. (2002): Feature selection for modular networks based on incremental training, *Journal of Intelligent Systems*, 13: 45–70.

GUAN, S-U. and ZHU, F. (2004): Modular feature selection using relative importance factors, *International Journal of Computational Intelligence and Applications*, 4: 57–75.

GUAN, S–U. and ZHU, F. (2004): Class decomposition for GA-based classifier agents – A Pitt approach, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34: 381–392.

JACOBS, R. A., JORDAN, M. I., NOWLAN, M. I. and HINTON, G. E. (1991): Adaptive mixtures of local experts, *Neural Computation*, 3: 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, in *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: 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.

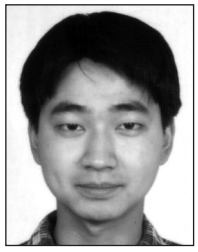
PRECHELT, L. (1997): Investigation of the CasCor family of learning algorithms, Neural Networks, 10: 885–896.

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

SQUIRES, C. S. and SHAVLIK, J. W. (1991): Experimental analysis of aspects of the cascade-correlation learning architecture, in Machine Learning Research GroupWorking Paper 91–1, Department of Computer Science, Univ.Wisconsin, Madison.

BIOGRAPHICAL NOTES

Chunyu Bao received the BSc and MSc degrees in physics from Beijing University, P.R. China, in 1999 and 2002, respectively. He is currently pursuing a PhD in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include machine learning, neural networks, pattern classification and clustering.

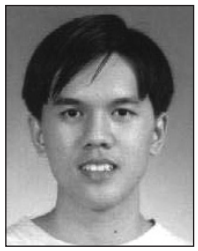


Chunyu Bao

Sheng-Uei Guan received his MSc and PhD from the University of North Carolina at Chapel Hill. He is currently a professor in the School of Engineering and Design at Brunel University, UK. Professor Guan 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 in Taiwan for three and half years. He served as deputy director for the Computing Center and the chairman for the Department of Information & Communication Technology. Later he joined the Electrical & Computer Engineering Department at the National University of Singapore as an associate professor.

Sheng-Uei Guan

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



TseNgee Neo

APPENDIX I

The Constructive Backpropagation algorithm (CBP) can be depicted briefly as follows (Guan and Li, 2003; Lehtokangas, 1999)

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:

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

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.

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 3. Adjust all the weights connected to the new unit (both input and output connections) by minimizing the modified sum of squared errors:

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

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

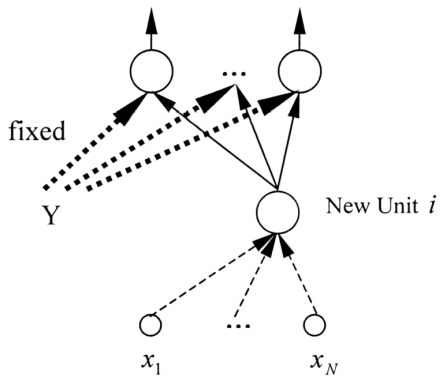


Figure 3: Training a new hidden unit in CBP learning.
Y represents previously added connections to network output units.

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 *i* th 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* (Squires and Shavlik, 1991), 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:

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

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*:

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

The *generalization loss* (Squires and Shavlik, 1991) at epoch *t* is defined as the relative increase of the validation error over the minimum so far (in percent):

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

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* (Squires and Shavlik, 1991) is defined to be a sequence of *m* epochs numbered $n + 1 \ldots n + \mathrm{m}$ where *n* is divisible by *m*. The training progress measured after a training strip is:

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

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 overfit or training has little progress. It is a bit unsatisfactory that all of these criteria are heuristic.