Low-Interference Output Partitioning for Neural Network Training

Shang Yang, Sheng-Uei Guan, Wei Fan Li, and Lin Fan Zhao

Abstract—**This paper presents a new output partitioning approach with the advantages of constructive learning and output parallelism. Classification error is used to guide the partitioning process so that several smaller sub-dimensional data sets are divided from the original data set. When training each sub- dimensional data set in parallel, the smaller constructively trained sub-network uses the whole input vector and produces a portion of the final output vector where each class is represented by one unit. Three classification data sets are used to test the validity of this algorithm, while the results show that this method is feasible.**

Index Terms—**Constructive learning algorithm, output partitioning, parallel growing, output interference**

I. Introduction

Neural networks [1]-[3], evolutionary algorithms [4], fuzzy logic [5], [6] and other methods have been proposed to tackle classification problems. Among them, neural networks based solutions have attracted much attention and become one of the most popular techniques for classification.

However, when neural-network is applied to real-world classification problems, it still suffers from some drawbacks, especially when used in large-scale problems. Internal interference exists during the training process [7], whenever updating the weights of hidden units the influence from two or more output units due to clash in their weight-updating directions[1].

The strategy "divide-and-conquer" is applied. The internal interference among outputs can be reduced by dividing the original problem into several sub-problems. However, several important issues are raised: how to divide the original problem into several smaller and simpler problems, how to assign a network module to learn each of the sub-problem, how to combine the individual modules into the solution to the original task. Up to now, there are several approaches to tackle these issues: functional modularity [8], domain decomposition, class decomposition [9], [10] and state decomposition [11].

At the same time, parallel training has also been used to gain faster training. By training several sub networks at the same time, the time spent on training can be greatly reduced [1].

Manuscript received December 28, 2012; revised February 22. 2013. This research is supported by the National Natural Science Foundation of China under Grant 61070085.

Shang Yang and Sheng-Uei Guanare are with the School of Electronic & Information Engineering, Xi'an Jiaotong University, Xi'an, China (e-mail: Steven.Guan@xjtlu.edu.cn)

Weifan Li and Linfan Zhao are with the Dept. of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China

For our work, we applied the output partitioning approach. A data set to be classified can be partitioned into several smaller sub-dimensional data sets with distinct classes. Each sub-dimensional data set is then handled by a smaller sub-network using the whole input vector as input and producing a portion of the final output vector. This method reduces computational time and improves performance.

In Section II, we briefly recall the constructive learning algorithm. The concept of output partitioning is described in Section III. The proposed partitioning algorithm is then described in Section IV. In Section V, experiments based on partitioning are implemented with results analyzed. Finally, the conclusions are presented in Section VI.

II. Constructive Backpropagation (CBP) Neural Network Algorithm

Constructive Learning Algorithm consists of Dynamic Node Creation method [12], Cascade-Correlation [13] as well as its variations [14]-[16], Constructive Single-Hidden-Layer Network [17] and Constructive Backpropagation [16] (CBP) and etc. For our work, CBP is used.

III. Sub-Group Model of Input Attributes

A. Output Attribute Group Model

All of the output attributes are partitioned into r sub-group containing at least one output:

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

$$=\sum_{p=1}^{P}\Big[\sum_{k_1=1}^{S_1}(o_{pk_1}-t_{pk_1})^2+\sum_{k_2=S_1+1}^{S_1+S_2}(o_{pk_2}-t_{pk_2})^2+\cdots+\sum_{k_r=S_1+S_2+\cdots+S_{r-1}+1}^{K}(o_{pk_r}-t_{pk_r})^2\Big]$$

$$=\sum_{p=1}^{P}\sum_{k_1=1}^{S_1}\left(o_{pk_1}-t_{pk_1}\right)^2+\sum_{p=1}^{P}\sum_{k_2=S_1 1}^{S_1+S_2}\left(o_{pk_2}-t_{pk_2}\right)^2+\ldots+\sum_{p=1}^{P}\sum_{k_r=S_1+S_2+\cdots+S_{r-1}+1}^{K}\left(o_{pk_r}-t_{pk_r}\right)^2$$

$$(S_1+S_2+\cdots+S_r=K) \tag{1}$$

Especially, E_1, E_2, …E_r are independent from each other. And the sum of them must be less than $E_{th.}$

B. Sub-Network Model

Sub-NN_1, sub-NN_2,…sub-NN_r replace the original network after grouping. These sub-networks are trained by CBP network. Each sub-NN produces only a portion of the result while the final result is generated by integrating the sub-networks by ensemble learning method.

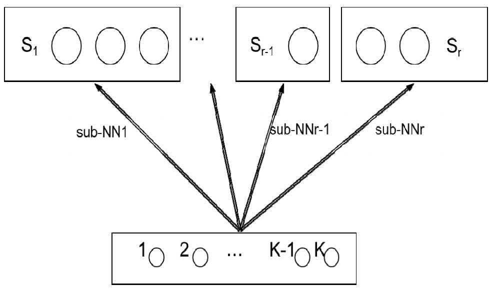


Fig. 1. Output attribute sub-network model

IV. Partitioning Algorithm based on Output Attributes

A. Definition

CBP neural networks are very sensitive to the change of training time. If training time is too short, the neural network won't be able to produce good result. However, long training will result in overfitting and poor bad generalization. In this article, the validation set is applied to determine the training time [18], [19].

A dataset is divided into three sub dataset: a training set is used to train the network; a validation set is used to evaluate the quality of the network to avoid overfitting during the training; finally, a test set is used to evaluate the resultant network. In this paper, the size of training, validation and test size is 50%, 25% and 25% of the dataset's total available patterns.

Training error E is mean square error percentage [18]. It is used to reduce the number of the coefficients in formula (1) and dependence on the range of output values.

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

In the above formula, o_{max} and o_{min} are the maximum and minimum output values in formula (1-1).

$E_{tr}(t)$ is per pattern's average error of trainingnetwork upon epoch t. $E_{ve}(t)$ is the corresponding error on validation set and is used to determine the time to stop training. $E_{te}(t)$ is the test error, used to describe the quality of the network. $E_{opt}(t)$ stands for the minimum validation error from the start to epoch t.

$$E_{opt}(t) = \min_{t' \le t} E_{va}(t') \qquad (3)$$

The relative increase of the validation error over the minimum so far is defined as the generalization loss at epoch t:

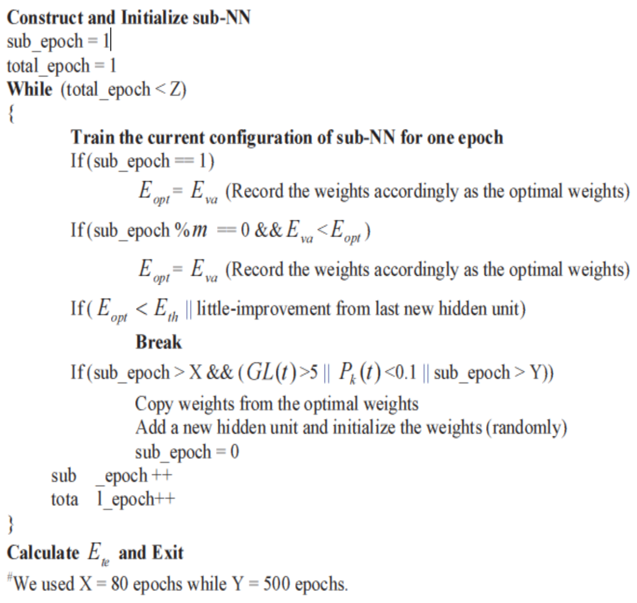
$$GL(t) = 100(\frac{E_{va}(t)}{E_{opt}(t)} - 1) \qquad (4)$$

Training will stop if the generalization loss is too high. Otherwise, it will result in overfitting. A training strip of length m [18] is defined as the sequence of m times repeating from n+1 to n+m. especially, n can be divided exactly by m. During the training strip, training progress is measured by $P_m(t)$ which means how much larger the average error is than the minimum.

$$P_m(t) = 1000(\frac{\sum_{t' \in t-m+1,\dots t} E_{tr}(t')}{m \min_{t' \in t-m=1,\dots t} E_{tr}(t')} - 1) \qquad (5)$$

B. Process for Sub-Network Growing and Training

Fig. 2 shows the procedure for sub-network growing and training.



```
Construct and Initialize sub-NN
sub_epoch = 1
total_epoch = 1
While (total_epoch < Z)
{
        Train the current configuration of sub-NN for one epoch
        If(sub_epoch == 1)
                $E_{opt} = E_{va}$ (Record the weights accordingly as the optimal weights)
        If(sub_epoch %$m$ == 0 && $E_{va} < E_{opt}$)
                $E_{opt} = E_{va}$ (Record the weights accordingly as the optimal weights)
        If( $E_{opt} < E_{th}$ || little-improvement from last new hidden unit)
                Break
        If(sub_epoch > X && ($GL(t)$>5 || $P_k(t)$<0.1 || sub_epoch > Y))
                Copy weights from the optimal weights
                Add a new hidden unit and initialize the weights (randomly)
                sub_epoch = 0
     sub _epoch ++
     tota l_epoch++
}
Calculate $E_{te}$ and Exit
#We used X = 80 epochs while Y = 500 epochs.
```

Fig. 2. The process for growing and training sub-networks

V. Output Partitioning Algorithm

When comparing with conventional large neural networks, employing several smaller sub-networks for learning tends to have lower classification errors as it can reduce the internal interference. The classification error is an important measurement of NN's performance. To obtain a lower classification error, an output partitioning algorithm, which employ several sub-network, was designed. We propose to get a near-optimal result via this algorithm. The details are presented as follows:

Step 1: Find the classification error C_i of each class and order them in ascending order as $\{C_a, \dots C_b, \dots C_c\}$, where $C_a < C_b < C_c$. To obtain the individualC_i, 1< i <K (K is the number of class), all patterns not belong to class i are labeled as patterns of class $\bar{i}$. A single NN is then used for the resulting two-class classification problem.

Step 2: Find the classification error of the every partition $\{i,j\}$ (1<i<k, $i \neq j$). And record the classification error as $C_{(i,j)}$.

Step 3: Judge the interference among every two classes according to the equation

$$F=C_{(i,j)}-(C_i+C_j) \qquad (6)$$

Which means that if F is negative the combination of (I, j) lower the classification error, vice versa.

Step 4: Pick up a class A in the sequence obtained in step1 and form a partition. If there is no group, create a new group and include A in that group. Else, iteratively find a group, in which all the contained class has no interference with A, and include A in that group. If failed to find such a group, build a new group and include that class. Delete A from the sequence.

Step 5: Repeat step4 until all until sequence obtained in step1 is empty.

VI. Experimental Results and Analysis

UCI machine learning datasets were used.

A. Glass

TABLE I: Glass Interference Matrix (Unit %)

Output NO.	1	2	3	4	5	6
1	20.28					
2	32.36	34.91				
3	24.25	33.58	8.30			
4	20.1	36.7	9.53	1.04		
5	22.65	34.91	9.91	2.36	0.85	
6	26.23	37.17	16.79	9.25	10.0	9.43

Each diagonal element represents the classification errors for each individual class while the rest represents the classification error of partition (*i*, *j*). Numbers in red represent results from interference-less pairs.

TABLE II: Glass Interference Table

Class NO.	**Classes without interference**	**Classes with interference**
1	2,3,4,6	5
2	1,3,5,6	4
3	1,2,6	4,5
4	1,6	2,3,5
5	2,6	1,3,4
6	**1,2,3,4,5**	

TABLE III: Experimental Results of Glass (Unit %)

	Partition result	Classification error
Non-partitioning		41.22
Full-partitioning	{1}{2}{3}{4}{5}{6}	36.13
Ascending order	{2,1,6,3}{4,5}	34.05
Descending order	{5,6}{4,1}{3,2}	32.25
Yinan Qi's[20]	{2,6,1}{3}{4}{5}	32.93
Random partitioning	{3,4}{2,6,5}{1}	36.15

Random partitioning means no strategy and order is applied.

According to the classification errors for individual classes, we can obtain two kinds of ordering: ascending and descending order. The ascending order is 2-1-6-3-4-5. The descending order is 5-4-3-6-1-2. So, we can get two different group by using the two orders: {2,1,6,3}{4}{5} and {5,6}{4,1}{3,2}.

B. Vowel

TABLE IV: Vowel Interference Table

Class Number	Classes without interference	Classes with interference
1	3,9,10	2,4,5,6,7,8,11
2	3,4,5,11	1,6,7,8,9,10
3	1,2,8,9,11	4,5,6,7,10
4	2,6,7,8,9,11	1,3,5,10
5	2,6,7,8,9,11	1, 3,4 ,10
6	4,5,8,11	1,2,3,7,9,10
7	4,5,8,9,10,11	1,2,3,6
8	3,4,5,6,7,9,10,11	1,2,
9	1,3,4,5,7,8,10,11	2,6
10	1,7,8,9,11	2,3,4,5,6
11	2,3,4,5,6,7,8,9,10	1

TABLE V: Vowel Interference Matrix (Unit %)

Class No.	1	2	3	4	5	6	7	8	9	10	11
1	2.00										
2	33.33	2.39									
3	3.52	3.46	2.00								
4	5.24	4.58	4.15	2.81							
5	9.76	6.72	7.92	9.47	5.43						
6	11.19	13.89	11.15	9.29	11.48	8.40					
7	11.58	9.25	7.35	6.66	8.12	14.07	4.72				
8	9.25	10.47	4.51	5.93	7.71	10.81	5.53	3.95			
9	8.52	12.23	6.90	6.90	10.90	16.48	11.64	8.18	7.45		
10	2.27	5.75	6.88	6.70	8.20	10.87	6.64	4.41	4.33	2.19	
11	9.37	7.69	7.49	8.54	10.77	14.86	11.30	9.09	7.65	7.35	6.84

TABLE VI: Experimental results of Vowel (Unit %)

	Partition result	Classification error
Non-partitioning		34.73
Full-partitioning		24.39
Ascending order	{6,11,5,8}{9,7,4}{2}{10,1}{3}	17.73
Descending order	{1,3,2}{10,8,7,11,9}{4,6}{5}	16.45
Yi'nan Qi [20]	{6,3,2,11,9}{10,1,8}{4,7}{5}	18.57
Random partitioning	{6,7,1,9}{4,8}{5,2,10,3}{11}	28.31

Random partitioning means no specific strategy and/or ordering is applied.

C. Thyroid

TABLE VII: Interference Matrix (Unit %)

Class NO.	1	2	3
1	1.58		
2	1.89	1.83	
3	1.90	1.98	1.72

Apparently, all the three outputs don't interference with one another. This means non-partitioning i.e. {1,2,3}.

TABLE VIII: Experimental Results of Thyroid (Unit %)

	Partition result	Classification error
Non-partitioning		1.86
Full-partitioning		1.89
Qi Yinan[20]	{2}{1,3}	1.72
Random partitioning	{3}{1,2}	1.88

Random partitioning means no strategy and/or ordering is applied.

VII. Conclusion

This paper presented a new approach for growing and training of neural network. By partitioning the output space, the performance of neural network is improved due to reduced interference. According to the experimental results of Glass and Vowel, this algorithm is better than full-partitioning, non-partitioning and the result from Yinan Qi et al [8]. We didn't get a good result for Thyroid and the possible reason is the small output number. However, the strategy would likely work on datasets with higher dimensions.

REFERENCE

[1] S. U. Guan and S. Li, "Parallel Growing and Training of Neural Networks Using Output Parallelism," *IEEE Trans ON Neural Networks*, vol. 13, no. 3, May 2002

[2] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, "Efficient classification for multiclass problems using modular neural networks," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 117–124, 1995.

[3] E. C. Paz, "Markov chain models of parallel genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 216–226, 2000.

[4] A. L. Corcoran and S. Sen, "Using real-valued genetic algorithm to evolve rule sets for classification," in *Proc. 1st IEEE Conf. on Evolutionary Computation*, Orlando, FL, 1994, pp. 120–124.

[5] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern.*, pt. Part B, vol. 29, no. 5, pp. 601–618, 1999.

[6] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 509–522, 2000

[7] R. A. Jacobs, M. I. Jordan *et al*., Adaptive mixtures of local experts, *Neural Comput*., vol. 3, no. 1, pp. 79–87, 1991.

[8] R. E. Jenkins and B. P. Yuhas, "A simplified neural network solutionthrough problem decomposition: The case of the truck backer-upper," *IEEE Trans. Neural Networks*, vol. 4, pp. 718–720, July 1993.

[9] 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 Networks*, vol. 10, pp. 1244–1256, Sept. 1999.

[10] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, "Efficient classification for multiclass problems using modular neural networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 117–124, Jan. 1995.

[11] V. Petridis and A. Kehagias, *Predictive Modular Neural Network: Applications to Time Series*, Boston, MA: Kluwer, 1998.

[12] T. Ash, "Dynamic node creation in back propagation networks," *Connection Sci*., vol. 1, 1989, pp. 365–375.

[13] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *Advances in Neural Information Processing systems*, vol. 2, 1990, San Mateo, CA: Morgan Kaufmann, pp. 524–532.

[14] L. Prechelt, "Investigation of the CasCor family of learning algorithms," *Neural Networks*, vol. 10, no.1997, pp. 885–896.

[15] S. Sjogaard, "Generalization in cascade-correlation networks," in *Proc. IEEE Signal Processing Workshop*, 1992, pp. 59–68.

[16] S. U. Guan and S. Li, "An approach to parallel growing and training of neural networks," in *Proc. 2000 IEEE Int. Symp. Intell.Signal Processing Commun. Syst.(ISPACS2000)*, Honolulu, HI.

[17] D. Y. Yeung, "A neural network approach to constructive induction," in *Proc. 8th Int. Workshop Machine Learning*, Evanston, IL, 1991.

[18] L. Prechelt, *A set of neural network benchmark problems and benchmarking rules, Technical Report 21/94*, Department of Informatics, University of Karlsruhe, Germany, 1994.

[19] L. Rechelt, "Investigation of the CasCor family of learning algorithms," *Neural Networks*, vol. 10, no. 5, pp. 885–896, 1997.

[20] S. U. Guan and Y. Qi, "Output partitioning of neural networks," *Neurocomputing*, vol. 68, 2005, pp.38–53.

[21] M. Lehtokangas, "Modeling with constructive backpropagation," *Neural Networks*, vol. 12, 1999, pp. 707–716.

[22] J. Ang, S. Guan, K. C. Tan *et al*., "Interference-less neural network training," *Neurocomputing*, vol. 71, no.16-18, pp. 3509-3524, 2008.

[23] M. H. Li, *Based on the low interference the integration study method of neural network*, Shanxi: Xi'an Jiaotong University, 2012.

Sheng-Uei Guan received his M.Sc. & Ph.D. from the University of North Carolina at Chapel Hill. He is currently a professor in the computer science and software engineering department at Xi'an Jiaotong-Liverpool University (XJTLU). He is also affiliated with Xi'an Jiaotong University as an adjunct faculty staff. Before joining XJTLU, he was a professor and chair in intelligent systems at Brunel University, UK.