Tree Architecture Pattern Distributor: A Task Decomposition Classification Approach

Victor M. O. Alves and George D. C. Cavalcanti

Abstract— **Task decomposition is a widely used method to solve complex and large problems. In this paper, it is proposed a novel task decomposition approach, named Tree Architecture Pattern Distributor (*TreeArchPD*), which is based on another task decomposition technique, called Pattern Distributor. The main idea is to design a tree architecture with many Distributors instead of using only one Distributor as proposed by the original technique. It is also proposed a new class grouping method that aims to optimize the class selection for task decomposition. Many experiments were done and they showed the effectiveness of the proposed approaches.**

MULTILAYERED feed-forward neural networks are widely used in the literature to solve diverse problems. It is widely known that these networks present some drawbacks when applied in real world problems. A common drawback is the network size. Small networks cannot learn the problem well, while large networks will lead to overfitting and thus poor generalization [2].

Various approaches were proposed to tackle these drawbacks, such as: pruning [3], regularization [4] and constructive algorithms [5], [6], [7]. Using these approaches the network size can be found in an automated way. However, there are real world problems that have many classes, many patterns and/or many features. For these problems, the approaches do not work satisfactory, because of the complexity of the training. In other words, large networks have a tendency to add internal interference into their input-to-hidden layer weights [15].

One alternative is to divide a problem into a set of smaller and simpler sub-problems, based on the divide-and-conquer methods, instead of using a single and large network. Various task decomposition methods have been proposed [8], [9], [10], [11]. Based on Lu and Ito [10] approach, Guan and Li [8] proposed the Output Parallelism (OP) method. Using OP, a problem can be divided into several sub-problems, each of which is composed by the whole input vector and a fraction of the output vector. The responsibility of each module is to produce a fraction of the output of the original problem [8]. Moreover, each module can be trained in parallel. However, some drawbacks appear in these methods, such as: too much time for training and manual class grouping.

Guan *et al.* [1] proposed a method called Pattern Distributor (PD). In this architecture there is a special module called Distributor which was introduced in the network architecture in order to improve the performance of the whole network. The Distributor works as a supervisor which makes high level classifications. Each pattern that is given as input to the Distributor is classified as belonging to one of the R modules. And each module is responsible to classifier a subset of the original classes of the problem. Therefore, when the number of modules increases the work of the Distributor becomes more difficult.

This paper proposes an approach to deal with the above mentioned weakness of the PD method. The idea is to create an automatic procedure to produce multiple Distributor modules which are arranged in a tree structure. Doing this, the original problem is divided into many sub-problems and consequently the task performed by each module is simplified. The proposed method is called *TreeArchPD* - Tree Architecture Pattern Distributor.

One important part of the *TreeArchPD* is an automatic procedure to group the classes. So, a novel approach to grouping the classes, based on the confusion matrix, is described. The proposed approach is compared with two approaches described in [1]: crosstalk table and genetic algorithm.

This paper is organized as follows. In Section I, some related works are presented. In Section II, the proposed method to find the best class grouping and the *TreeArchPD* method are presented. Section III shows the experiments, results and discussion. Finally, the conclusions are presented in Section IV.

I. Related Works

Task decomposition approach is known in the literature for quite some time. Anand *et al*. [9], in 1995, proposed the use of a modular neural network, decomposing a K-class problem into K 2-class sub-problems. Each K module (representing one of K sub-problem) is trained to learn only one sub-problem. However, each module is used in the classification phase to discriminate patterns belonging to other sub-problems. The final decision is made based on the maximum rule.

Lu and Ito [10] proposed a method that divides a K-class problem into C_k^2 2-class sub-problems (C stands for combinations or choices). But in this case the training is done for each module, only with its sub-problem patterns. A proposed method called *min-max* modular network was used as a method to decompose the problem classes according to its relations and to combine the K-modules responses.

Lu *et al.* [11] proposed an architecture that consists into several cascade neural networks. Each network is responsible

Victor M. O. Alves and George D. C. Cavalcanti are with the Center of Informatics (CIn), Federal University of Pernambuco (UFPE), Brazil, {vmoa,gdcc}@cin.ufpe.br. Site: www.cin.ufpe.br/∼viisar

for classification, but more roughly at the beginning and gradually soften towards the end.

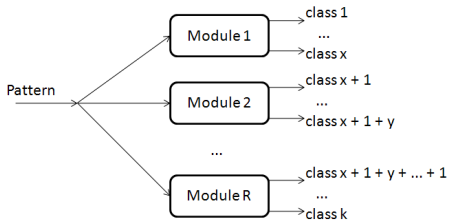

Fig. 1. Output Parallelism architecture with R modules.

Guan and Li [8] proposed a method named Output Parallelism (OP). In this method, a K-class problem is divided into R sub-problems; see Fig. 1 for details. The notation used by the author is the same one used in this paper. Suppose a problem with 6 classes. If the problem is divided into three sub-problems with 2 classes each, the notation used to represent this configuration is {{class 1, class 2}, {class 3, class 4}, {class 5, class 6}}. Each module is trained with the whole training set. When an unseen pattern is presented into an OP network, it is processed by all the modules and the final result is obtained by combining the results. This method allows a parallel modular processing, optimizing the training.

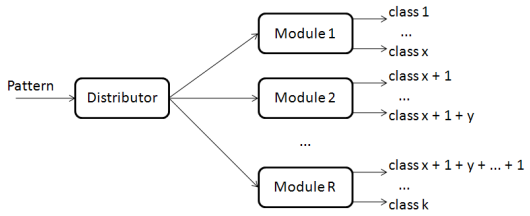

Fig. 2. Pattern Distributor architecture with R modules.

Guan *et al.* [1] modified the OP method, creating the Pattern Distributor (PD) method. In the PD method, as well as in the OP method, a K-class problem is divided into R sub-problems. However, in this experiment another module is added, called Distributor; see Fig. 2 for details. The Distributor module is responsible for sending a pattern to the corresponding module non-distributor (here called specialist). Thus, in the PD method, after training, an unknown pattern is processed by two modules: the Distributor and one of the R specialists which is chosen by the Distributor. Based on this fact, the authors proposed the Reduced Pattern Training (RPT) technique. It consists in training each specialist module only with the patterns belonging to its sub-problem.

One drawback of these task decomposition methods is the class grouping. Reliable and automatic methods are required for the success of these methods. Guan *et al.* [1] indicated two approaches to automate the class selection: crosstalk table and genetic algorithm.

Crosstalk table uses Fisher linear discriminant to project the pattern feature vectors into 1-D space. After that, the distances between all patterns are computed. These distances are arranged to form a table which is called crosstalk table. Classes with the smaller distance are grouped.

Another alternative is to use a genetic algorithm to find an optimal or a sub-optimal combination through evolution. The authors proposed a specific chromosome to this problem; details in [1].

Freitas *et al.* [12] defined manually the class grouping using the confusion matrix (CM). In order to have the first CM, they trained an ordinary single neural network. After the analysis of the generated CM, the modular network was designed.

II. New Task Decomposition Architecture

As has been briefly shown, the PD method presents a weakness when the problem is large, i.e., a problem with many classes. In this case, there are two possibilities: the number R of modules can be large or small.

If the number R is small, the task of the Distributor is much more simple. However, each module (specialist) has a difficult problem to solve. Conversely, if R is a large number, the Distributor has a big problem to solve. While the work of the specialists is not complex. In both cases, an undesirable fact emerges: the classification error increases.

Therefore, an investigation about how to find the best number of modules, aiming to increase the accuracy rate of the whole system, is of great importance. Two approaches to perform this task were previously proposed in [1], as presented in the end of Section I. However, the best one is based on genetic algorithm, which is a very expensive alternative, in terms of computational time. So, in the following Sections II-A and II-B are presented a new and automatic class grouping for task decomposition and the proposed *TreeArchPD* method, respectively.

A. Proposed Class Grouping Method

Before task decomposition, it is necessary to find the optimal or sub-optimal class groups which aim to facilitate the work of the Distributor and the work of the modules. So, it was developed a new automatic method which extracts information from the confusion matrices. Freitas *et al.* [12] showed that it is possible to arrange the similar classes based on the confusion caused by a classifier. Table I shows a hypothetic confusion matrix of a 8-class problem. Note that Class 1 and 3 are quite confused. And Class 2 and 6 have confusion too. The same can be observed in other classes. From this point of view, it is natural to say that Class 1 and 3 are similar. The same can be said to Class 2 and 6. Then, basically, the method analyzes the most confused classes, grouping them.

Inspired on this idea, an automatic method was developed. For a confusion matrix CM, the distance matrix D is calculated using Equation 1.

$$D_{i,j} = \begin{cases} 0, \text{ if } i = j, \\ 1 - (nCM_{i,j} + nCM_{j,i}), \text{ otherwise.} \end{cases} \quad (1)$$

TABLE I

HYPOTHETIC CONFUSION MATRIX.

	Predicted							
Real	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	180	0	20	0	0	0	0	0
2	0	195	0	0	0	5	0	0
3	10	0	190	0	0	0	0	0
4	0	0	0	175	25	0	0	0
5	0	0	0	12	188	0	0	0
6	0	8	0	0	0	192	0	0
7	0	0	0	0	0	0	193	7
8	0	0	0	0	0	0	22	178

From Equation 1, i and j represent the confusion matrix indexes. First of all, the matrix is normalized into $[0 \dots 1]$ interval. Each matrix element, $CM_{i,j}$ is divided by the sum of its line, i.e., $nCM_{i,j} = CM_{i,j} / \sum_{j=1}^{c} CM_{i,j}$, where c is the number of columns. After that, the distance matrix D is calculated according to the number of confusions per class. Thus, when $D_{i,j}$ has a small value, this means that there is a high probability of confusing classes i and j ($i \neq j$).

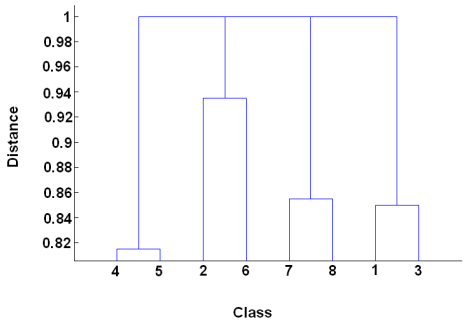

Fig. 3. A dendrogram generated based on the confusion matrix shown in Table I.

Afterward, a hierarchical clustering algorithm is applied to the distance matrix in order to group the most similar classes. The algorithm used was Unweighted Pair Group Method with Arithmetic Mean [13]. A dendrogram was generated based on the confusion matrix (Table I), it is shown in Fig. 3.

B. Proposed Task Decomposition Architecture

Guan *et al.* [1] did a series of experiments which aimed to demonstrate the Pattern Distributor method. In that work, the authors suggested, as future work, to develop an architecture in which two or more Distributors were arranged. Therefore, the objective of this section is to show an extension of the Pattern Distributor method that deals with more than one Distributor.

The methodology to incorporate into the system more Distributor modules is as follows. Suppose the same 8-class problem showed in Section II-A. First, using the class decomposition procedure proposed (Section II-A), the amount of non-distributor modules (specialist modules) is determined by the dendogram generated; see Fig. 4. Each rectangle in this figure represents a module.

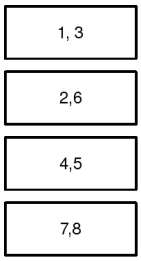

Fig. 4. Specialists modules organized by the class grouping algorithm.

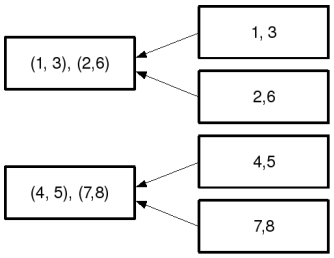

Fig. 5. Specialists modules forming its fathers.

Considering each specialist as a leaf, the next step is to group the specialist classes, forming a root for each pair; see Fig. 5. This step is repeated for all modules and is based on the dendrogram.

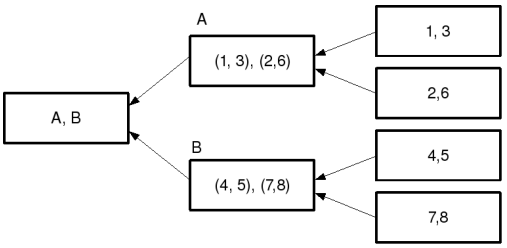

Fig. 6. Final architecture.

Only three layers are considered. So, the final step is to combine all the intermediary roots into one big root, as can be seen in Fig. 6.

This PD variation recalls a tree structure, so it is called *TreeArchPD*. Then, any unseen pattern will be evaluate by, not more than, 3 (three) modules. Algorithm 1 summarizes the whole procedure of the *TreeArchPD*.

III. EXPERIMENTS AND DISCUSSION

In order to illustrate the efficiency of the proposed methods, two experiment were made. In the first one, the proposed class grouping approach is evaluated. For comparison purposes, two databases that were used in [1] were chosen. In the second experiment, which aims to validate the proposed *TreeArchPD*, two more databases were chosen. All databases were obtained from the UCI Machine Learning Repository [14].

All training method mentioned here use *Resilient Back propagation* with learning rate = 0.1, maximum number of epochs = 1000, and the number of hidden units was chosen

Algorithm 1 TreeArchPD

1: Evaluate the data set using an ordinary classifier (here, a MLP was used);
2: From this evaluation a confusion matrix is obtained;
3: Normalize the confusion matrix values into $[0 \dots 1]$ interval:

$$nCM_{i,j} = CM_{i,j} / \sum_{j=1}^{c} CM_{i,j},$$

where nCM is the normalized matrix and c is the number of columns.
4: Calculate the distance matrix D, as stated in Equation 1:

$$D_{i,j} = \begin{cases} 0, \text{ if } i = j, \\ 1 - (nCM_{i,j} + nCM_{j,i}), \text{ otherwise.} \end{cases}$$

5: Matrix D is given as input to a hierarchical clustering algorithm: Unweighted Pair Group Method with Arithmetic Mean [13];
6: Based on the answer of the clustering algorithm, the leaves of the *TreeArchPD* are created.
7: The most similar leaves are combined into groups, based on the hierarchical clustering tree, forming its roots.
8: These roots are combined in a new and principal root. A restriction: no more than three layers of modules must be created;
9: All modules (PDs and Specialists) are trained independently.

in preliminary tests, it ranges between 50 and 150. The best configuration was chosen to this paper.

A. Databases

Four databases were used in this paper: *Segmentation*, *Letter*, *Optdigits* and *Vehicles*. The main features of these databases are described below.

1) Segmentation: This data set is formed by image segmentation data. It consists of 18 inputs, 7 outputs and 2,310 patterns. In this paper, 1,155 training, 578 validation and 577 test patterns were used.

2) Letter: This data set presents character image features. It consists of 16 inputs, 26 outputs and 20,000 patterns. In this paper, 10,007 training, 5,029 validation and 4,964 test patterns were used.

3) Optdigits: This data set presents numerical character image features. It consists of 64 inputs, 10 outputs and 5,620 patterns. In this paper, 2,811 training, 1,418 validation and 1,391 test patterns were used.

4) Vehicles: This data set presents vehicles silhouette features. It consists of 18 inputs, 4 outputs and 946 patterns. In this paper, 473 training, 238 validation and 236 test patterns were used.

The attributes of all databases were normalized into $[0 \dots 1]$ interval.

B. Class Grouping

The experiments described in this section show that the proposed approach to perform class grouping is better or, at least, equivalent to the Guan's methods. Besides that, it is important to mention that the proposed method is faster than PD method, because of the simplicity of the modules. It was used two databases: *Segmentation* and *Letter*. The same databases were used in [1], where it was proposed two approaches to find the best class clusters: genetic algorithm and crosstalk table. These two approaches should be used for comparison reasons.

The proposed method was used twice for each database, with different number of groups. In the following subsections, the two experiments are described and compared with the experiments found in [1].

1) Experiments with the Segmentation data: Two combinations were obtained using the crosstalk based combination of classes [1]: {{1,3,4,5}, {2,6}, {7}} and {{1,2,7}, {3,4}, {5,6}}. This class grouping reached 4.6187% and 5.39% of classification error, respectively. Using a genetic algorithm, it was obtained the combination [1]: {{1,3,4,5}, {6,7}, {2}} and achieved 4.5321% of error rate.

Before using the proposed method, an ordinary *Multilayer Perceptron* with 18 inputs, 7 outputs and one hidden layer with 120 hidden units was trained. From the confusion matrix, the proposed method obtained two combinations: {{1,3,4,5}, {2,6,7}} and {{1,3,4,5}, {6,7}, {2}}. The second combination was identical to the GA combination. The combinations reached 5.3726% and 4.9567% of classification error, respectively. The results are shown in Table II.

TABLE II
EVALUATING THE CLASS GROUPING STRATEGY - SEGMENTATION DATA

	Error (%)	
Grouping method	**Distributor**	**Overall**
Crosstalk-based 1 [1]	0.1040	4.6187
Crosstalk-based 2 [1]	4.7834	5.3900
GA-based [1]	0.0173	4.5321
Proposed (2 groups)	0.3986	5.3726
Proposed (3 groups)	0.0867	4.9567

From Table II, crosstalk-based 1 and 2 refer to the two combinations generated by crosstalk table in [1]. GA-based refer to combination obtained by Guan, using genetic algorithm. Finally, the proposed method was used twice, with 2 and 3 groups. The Distributor Error is the error obtained only in the Distributor module. The Overall error is the error of the whole system.

Table II shows that apparently the GA-based method presented the best rates. However, the proposed method (with 3 groups) obtained the same class grouping that the GA method. In other words, the difference was how the modules were trained, once they have the same class grouping. And it is important to mention that the proposed method was faster than the genetic algorithm. According to [1], the GA method takes hours to execute. The proposed one takes few seconds.

2) Experiments with the Letter data: The classes were manually grouped in [1], resulting in: {{1,2,3,4,5,6,7}, {8,9,10,11,12,13,14}, {15,16,17,18,19,20}, {21,22,23,24,25,26}}, which yielded 15.855% of error. Neither crosstalk-based or GA-based combination were used.

TABLE III
EVALUATING THE CLASS GROUPING STRATEGY - LETTER DATA

	Error (%)	
Grouping method	**Distributor**	**Overall**
Manually (6 groups) [1]	12.1950	15.8550
Proposed (4 groups)	3.7893	13.5133
Proposed (6 groups)	7.7337	12.8223

Before using the proposed method, an ordinary *Multilayer Perceptron* with 16 inputs, 26 outputs and one hidden layer with 80 hidden units was trained. From the confusion matrix, the proposed method created two class combinations: {{2,3,4,5,7,8,10,11,12,14,15,17,18,19,24,26}, {6,9,16,20,25}, {13,21,22,23}, {1}} and {{2,4,7,8,14,15,17,18}, {3,5,10,11,12,19,24,26}, {20,25}, {6,9,16}, {1}, {13,21,22,23}}. These combinations yielded, respectively, 13.5133% and 12.8223% of classification error. Table III shows the results.

For this data set, Guan did not used an automatic class grouping. So, Table III only presents the results for the manual selection of the classes. It is clear that the proposed method outperforms the manual selection. The Distributor error achieved lower error rates when 4 groups were used. The explanation is the following: for 6 groups, the Distributor module have to distinguish between 6 classes, rather than 4 classes, making it more susceptible to errors.

These two experiments showed the usefulness of the proposed method. The results for the Segmentation data were similar to the results found in [1], and the results for the Letter data were better than the methods in [1]. Also, the proposed method is faster than the genetic algorithm approach.

C. Task Decomposition

The usefulness of the confusion matrix based method was previously shown. This section describes the experiments with the proposed PD method extension, named *TreeArchPD*. Therefore, four databases are used and a comparison with the original PD method is made.

To compare the PD method with the *TreeArchPD* method, four experiments were made; details of the experiments are exposed in the sections below. All the class grouping processes were made using the confusion matrix method described in Section II-A. All experiments were executed 10 times, varying weights initialization. Mean, standard deviation and training time are presented. Note that the training time does not take into account the class grouping process and the training was done sequentially. In the result tables, the final error rate is the overall error. The error rate per module was obtained considering only the patterns that arrived correctly to the modules.

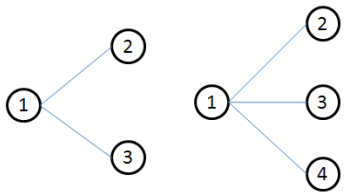

Fig. 7. PD architecture used for the Segmentation data set. At the left, 2 specialist modules and at the right 3 specialists. Number "1" represents the Distributor.

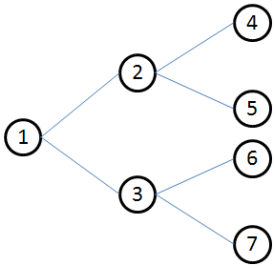

Fig. 8. TreeArchPD task decomposition architecture for Segmentation data set.

1) Experiments with the Segmentation data: The first experiment evaluates the performance of an ordinary MLP in this data set. A MLP with 7 outputs and 120 hidden units was used. Next, two architectures of PD method were used, one with 2 specialist modules and another with 3 specialists, both with 120 hidden units. Lastly, the *TreeArchPD* method was used, also with 120 hidden units for each module. Figs. 7 and 8 depict the architectures used to the PD method and the *TreeArchPD* method, respectively.

The class grouping for the PD method, with 2 specialists, was: module 2 = {1,3,4,5} and module 3 = {2,6,7} (see Fig. 7 left). Using 3 specialists, the class grouping was: module 2 = {2}, module 3 = {6,7} and module 4 = {1,3,4,5} (see Fig. 7 right). For the proposed method (see Fig. 8), the class grouping was: module 2 = {1,3,4,5}, module 3 = {2,6,7}, module 4 = {1,3}, module 5 = {4,5}, module 6 = {2,6} and module 7 = {7}.

TABLE IV
SEGMENTATION DATA RESULTS: CLASSIFICATION ERROR (%) AND TRAINING TIME (SECONDS)

Method	**Error $\overline{x}(\sigma)$**	**Training time**
MLP	6.7764 (0.8624)	114.00
PD (2 specialists)	5.3726 (0.7532)	151.52
PD (3 specialists)	4.9567 (0.6747)	116.73
TreeArchPD	5.3899 (0.6456)	173.26

Table IV shows the classification error and the training time for all approaches. The standard deviations are shown in brackets. Table V shows the classification error for each module separately (see Figs. 7 and 8). For this data set, the

TABLE V

SEGMENTATION DATA RESULTS: MODULAR ERROR (%)

Module	PD (2 spec.)	PD (3 spec.)	TreeArchPD
1	0.3986	0.0867	0.3640
2	8.8036	0	7.5319
3	0	0	0.7165
4	–	8.5781	8.8862
5	–	–	8.9786
6	–	–	1.0332
7	–	–	0

PD method achieved a better absolute error rate than the proposed method.

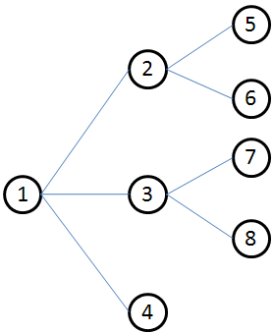

Fig. 9. TreeArchPD task decomposition architecture for Optdigits data set.

2) Experiments with the Optdigits data: For this data set, a MLP with 80 hidden units and 10 outputs was trained. The two configurations of the PD method were identical to the Segmentation data (Fig. 7). However, each module was trained with 80 units in the hidden layer. The *TreeArchPD* method has 8 modules, with 80 hidden units each. Fig. 9 shows the proposed architecture for the *TreeArchPD*.

The two configurations of PD (see Fig. 7) presented, respectively, the following class grouping: {{5,7}, {1,2,3,4,6,8,9,10}} and {{2,8,9,10}, {1,3,4,6}, {5,7}}, representing {{module 2}, {module 3}} and {{module 2}, {module 3}, {module 4}}, respectively. The *TreeArchPD* class distribution was: module 2 = {2,8,9,10}, module 3 = {1,3,4,6}, module 4 = {5,7}, module 5 = {2,8}, module 6 = {9,10}, module 7 = {3,4} and module 8 = {1,6}.

Table VI shows the classification error (mean and standard deviation) and training time (in seconds) for all the classifiers evaluated. The isolated module classification error is shown in Table VII.

TABLE VI

OPTDIGITS DATA RESULTS: CLASSIFICATION ERROR (%) AND TRAINING TIME (SECONDS)

Method	Error $\overline{x}(\sigma)$	Training time
MLP	5.4062 (0.4532)	50.00
PD (2 specialists)	5.2121 (0.4760)	79.00
PD (3 specialists)	5.1833 (0.2783)	53.54
TreeArchPD	4.7664 (0.4367)	61.91

TABLE VII

OPTDIGITS DATA RESULTS: MODULAR CLASSIFICATION ERROR (%)

Module	PD (2 spec.)	PD (3 spec.)	TreeArchPD
1	0.6732	2.3580	2.1639
2	0.9284	4.0996	5.7767
3	5.4183	2.7098	2.1389
4	–	0.8436	4.7565
5	–	–	9.1070
6	–	–	3.9655
7	–	–	2.3779
8	–	–	3.5693

In Table VI, it is presented the error of the MLP, the *TreeArchPD* and the two configurations of PD method. Note that, for this data set, the proposed method outperforms the other ones in terms of the accuracy rate. For training time, the proposed one was faster than PD method with two specialists.

3) Experiments with the Vehicles data: For this data set, a MLP with 50 hidden units and 4 outputs were used. This data set has only 4 classes, for that reason, the PD method and the *TreeArchPD* method had the same architecture. This architecture is identical to the one shown in Fig. 7 at left. The class distribution was: module 2 = {2,4} and module 3 = {1,3}. The classification error and the training time can be observed in Table VIII. The modular error is shown in Table IX.

TABLE VIII

VEHICLES DATA RESULTS: CLASSIFICATION ERROR (%) AND TRAINING TIME (SECONDS)

Method	Error $\overline{x}(\sigma)$	Training time
MLP	28.6893 (3.4663)	8.00
TreeArchPD	27.8155 (2.5073)	22.51

TABLE IX

VEHICLES DATA RESULTS: MODULAR CLASSIFICATION ERROR (%)

Module	TreeArchPD
1	8.3495
2	45.1927
3	10.0773

The proposed method and the PD method have the same structure, for this data set. They have the same classification error, which is better than MLP. For problems with few classes, the PD method is a specialization of the *TreeArchPD*.

4) Experiments with the Letter data: For this data set, a MLP with 80 hidden units and 26 outputs was used. For the PD method, it was used 4 and 6 specialist modules (see Fig. 10); each module with 80 hidden units. For *TreeArchPD*, motivated by the large number of classes, two configurations were used. One configuration with no more than two leaves for each intermediate layer node (Fig. 11 left) and other one with more leaves per intermediate layer node (Fig. 11 right).

The classes distribution in the PD methods are presented in Table X. For the *TreeArchPD* method, the distribution is shown in Table XI.

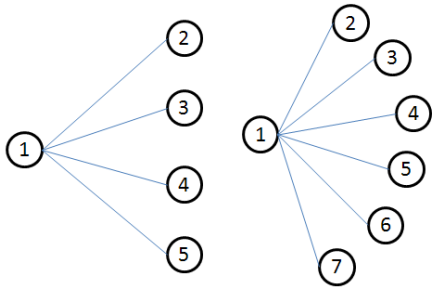

Fig. 10. PD task decomposition architecture for Letter data set. Left, with 4 specialists. Right, with 6 specialists.

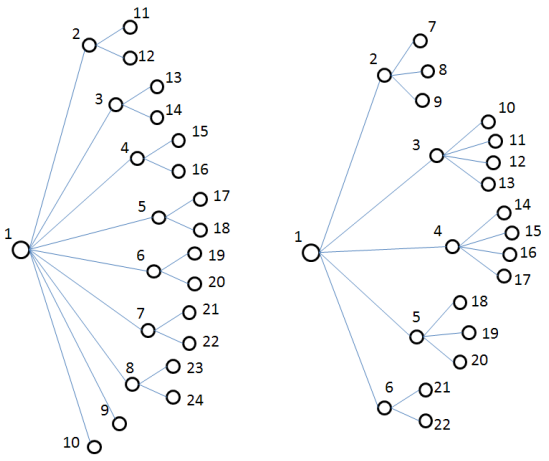

Fig. 11. TreeArchPD configurations for the Letter recognition problem. Configuration 1 (left), with no more than two leaves into the intermediate layer and Configuration 2 (right), with more leaves in the intermediate layer.

TABLE X

PD CLASS DISTRIBUTION FOR THE LETTER DATA.

Module	PD (4 specialists)	PD (6 specialists)
2	{2,3,4,5,7,8,10,11,12, 14,15,17,18,19,24,26}	{2,4,7,8,14,15,17,18}
3	{6,9,16,20,25}	{3,5,10,11,12,19,24,26}
4	{13,21,22,23}	{20,25}
5	{1}	{6,9,16}
6	-	{1}
7	-	{13,21,22,23}

The classification error, the standard deviation and the training time are shown in Table XII. Table XIII and Table XIV show the error per module.

Table XII shows that the PD and the TreeArchPD techniques outperformed the ordinary MLP. Configuration 1 of the proposed method presented the worst error rate, excepting the MLP. One possible explanation relies on the fact that there are a great number of modules in the intermediate layer (see Fig. 11 left), and this may cause higher error rate in the first module layer. However, TreeArchPD - Configuration 2 - presented similar classification error rate when compared with the PD (6 specialists). Thus, given the low standard deviation, the proposed method (Configuration 2) has achieved very good results for this data set.

TABLE XI

TREEARCHPD CLASS DISTRIBUTION FOR THE LETTER DATA.

Module	TreeArchPD (conf. 1)	TreeArchPD (conf. 2)
2	{5,19,24}	{3,5,11,19,24}
3	{3,10,11,26}	{1,2,10,12,18,26}
4	{2,12,18}	{4,7,8,14,15,17}
5	{4,8,14}	{6,9,16,20,25}
6	{7,15,17}	{13,21,22,23}
7	{6,9,16}	{19,24}
8	{13,20,23,25}	{5}
9	{21,22}	{3,11}
10	{1}	{10,26}
11	{19,24}	{12}
12	{5}	{2,18}
13	{3,11}	{1}
14	{10,26}	{14}
15	{12}	{4,8}
16	{2,18}	{7,17}
17	{14}	{15}
18	{4,8}	{6,16}
19	{7,17}	{9}
20	{15}	{20,25}
21	{6,16}	{13,23}
22	{9}	{21,22}
23	{20,25}	-
24	{13,23}	-

TABLE XII

LETTER DATA RESULTS: CLASSIFICATION ERROR (%) AND TRAINING TIME (SECONDS)

Method	Error $\overline{x}(\sigma)$	Training time
MLP	18.0459 (0.5136)	969
PD (4 specialists)	13.5133 (0.5861)	1403
PD (6 specialists)	12.8223 (0.8221)	1356
TreeArchPD (conf. 1)	14.5870 (0.6507)	1225
TreeArchPD (conf. 2)	13.0077 (0.1969)	1461

TABLE XIII

LETTER DATA RESULTS: MODULAR CLASSIFICATION ERROR OF PD METHOD (%)

Module	PD (4 spec.)	PD (6 spec.)
1	3.7893	7.7337
2	14.8679	8.9853
3	3.4352	6.1889
4	1.3630	1.4710
5	0	3.3124
6	–	0
7	–	2.1093

IV. CONCLUSIONS

Various task decomposition approaches have been developed to solve real world problems. These approaches use divide-to-conquer methods to reduce the problem complexity. Two examples of these task decomposition techniques are: Output Parallelism [8] and Pattern Distributor [1]. To choose the class grouping for the problem division, these articles proposed the use of crosstalk table and genetic algorithm. However, the genetic algorithm is very slow, as affirmed by Guan *et al.* [1].

The contribution of this paper is twofold. First, a new

TABLE XIV

LETTER DATA RESULTS: MODULAR CLASSIFICATION ERROR OF THE TREEARCHPD METHOD (%)

Module	TreeArchPD (conf. 1)	TreeArchPD (conf. 2)
1	12.8807	12.8807
2	20.8861	20.8861
3	11.8142	11.8142
4	18.9267	18.9267
5	16.5335	16.5335
6	19.4234	19.4234
7	6.4388	6.4388
8	9.5986	0
9	21.9918	9.5986
10	0	21.9918
11	15.0344	0
12	0	15.0344
13	9.3338	0
14	24.5333	0
15	0	9.3338
16	23.0862	24.5333
17	0	0
18	20.9698	23.0862
19	9.7123	0
20	0	20.9698
21	9.6602	9.7123
22	0	9.6602
23	11.0308	–
24	6.6642	–

method for class grouping was proposed which is based on the confusion matrices. The experiments showed its effectiveness. Second, an extension of the method proposed by Guan was presented, called *TreeArchPD*. The experiments showed that this extension outperformed the original PD method. In terms of computational time, the proposed approach is faster than other similar methods, especially when the training of the modules is performed in parallel.

The experiments showed the viability of the class grouping method and the TreeArchPD method. As future work, more experiments and configurations for the TreeArchPD must be evaluated.

ACKNOWLEDGMENTS

This work was supported in part by the Brazilian National Research Council CNPq (Proc. 475911/2008-3) and by FACEPE - Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (Proc. APQ-0890-1.03/08).

REFERENCES

[1] S. U. Guan, C. Bao and T. Neo, *Reduced Pattern Training Based on Task Decomposition Using Pattern Distributor.* IEEE Transactions on Neural Networks, vol. 18, pp. 1738-1749, 2007.

[2] S. U. Guan, Q. Yinan, S. K. Tan and S. Li, *Output partitioning of neural networks.* Neurocomputing, vol. 68, pp. 38-53, 2005.

[3] R. Reed, *Pruning algorithm: a survey.* IEEE Transactions on Neural Networks, vol. 4, pp. 740-747, 1993.

[4] T. Poggio, F. Girosi, *Regularization algorithms for leaning that are equivalent to multi-layer networks.* Science, vol. 247, pp. 978-982, 1990.

[5] T. Ash, *Dynamic node creation in backpropagation networks.* Connect. Sci., vol. 1, pp. 365-375, 1989.

[6] L. Prechelt, *Investigation of the CasCor family of learning algorithms.* Neural Networks, vol. 10, pp. 885-896, 1997.

[7] M. Lehtokangas, *Modelling with constructive backpropagation.* Neural Networks, vol. 12, pp. 707-716, 1999.

[8] S. U. Guan and S. C. Li, *Parallel growing and training of neural networks using output parallelism.* IEEE Transactions on Neural Networks, vol. 13, pp. 542-550, 2002.

[9] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, *Efficient classification for multiclass problems using modular neural networks.* IEEE Transactions on Neural Networks, vol. 6, pp. 117-124, 1995.

[10] B. L. Lu and M. Ito, *Task decomposition and module combination based on class relations: A modular neural network for pattern classification.* IEEE Transactions on Neural Networks, vol. 10, pp. 1244-1256, 1999.

[11] B. L. Lu, H. Kita, and Y. Nishikawa, *A multisieving neural-network architecture that decomposes learning tasks automatically.* Proc. IEEE Conference on Neural Networks, Orlando, FL, pp. 1319-1324, 1994.

[12] C. A. O. Freitas, L. S. Oliveira, F. Bortolozzi and S. B. K. Aires, *Handwritten Character Recognition Using Nonsymmetrical Perceptaul Zoning.* International Journal of Pattern Recognition and Artificial Intelligence, vol. 21, pp. 1-21, 2007.

[13] L. Kaufman and P. Rousseeuw, *Finding Groups in Data.* Wiley, 1990.

[14] Asuncion, A. and Newman, D. J., *UCI Machine Learning Repository.* Irvine, CA: University of California, School of Information and Computer Science, 2007.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, *Adaptive mixtures of local experts*, Neural Computation, vol 3, pp. 7987, 1991.