An Algorithm for Pruning Redundant Modules in Min-Max Modular Network

Hui-Cheng Lian and Bao-Liang Lu
Department of Computer Science and Engineering, Shanghai Jiao Tong University
1954 Hua Shan Rd., Shanghai 200030, China
Email: {lianhc,blu}@cs.sjtu.edu.cn

***Abstract*— The min-max modular (M^3) network is a framework that is capable of solving large-scale pattern classification problems in a parallel way. The M^3 network has been successfully applied to several large-scale real-world prblems. When a complex problem is decomposed into a number of separable problems, however, the M^3 network sufferes from its high redundancy of individual modules. This paper proposes an algorithm, called back-searching (BS) algorithm, to prune these redundant modules. The main idea behind the BS algorithm is to use the actual outputs of the trained M^3 network associated with training data to find out the redundant modules by means of 'back searching' . In order to ensure the correctness of the algorithm, we prove two propositions theoretically, namely the sufficient proposition and the necessary proposition, and perform simulations on several benchmark and real-world problems. The simulation results indicate that most of all redundant modules can be pruned by our proposed algorithm and the pruned network has the same generalization performance as the original network.**

I. Introduction

The min-max modular (M^3) network [1][2] is a framework that is capable of solving large-scale pattern classification problems in parallel. In comparison with other modular neural network models and machine learning approaches, the M^3 network has the following two important advantages. a) The task decomposition scheme is more general than one-vs-one scheme, which will be referred to as the "part-vs-part'"or PVP scheme throughout this paper. By using the PVP scheme, any user can decompose a complex multiclass problem into many two-class subproblems as samll as needed. Neither domain specialists nor prior knowledge of the problem is required. Since each of the two-class subproblems can be treated as a completely separable classification problem in the learning phase, all of the two-class subproblems can be learned in a parallel way. b) After learning each of the two-class subproblems with a network module, all trained network modules can be integreted into a M^3 network automatically according to two module combination rules, namely the minimization pricinple and the maximization principle [2].

In the last few years, the M^3 network has been successfully applied to many large-scale real-world problems such as part-of-speech tagging [5], single-trial EEG signal classification [6], and text categorization [8]. In M^3 network, a K-class classification problem is decomposed into a series of $K(K-1)/2$ two-class problems. These two-class problems are to discriminate class C_i from C_j for $i = 1, ..., K-1$ and $j = i+1, ..., K$, while the existence of the training data belonging to the other K-2 classes is ignored. If these two-class problems are still hard to be learned, they can be divided into a set of two-class subproblems as small as needed. Consequently, a large-scale and complex K-class classification problem can be solved effortlessly and efficiently by learning a series of smaller and simpler two-class problems in a parallel way. In order to make the framework more clear, we show the M^3 network in Fig. 1. This M^3 network include three parts: individual network modules, MIN units and MAX units (see Fig. 1). Here, the basic function of an MIN unit is to find a minimum value from its multiple inputs and the basic function of an MAX unit is to find a maximum value from its multiple inputs, and the individual network module can be any classifier such as MLP and SVM.

According to the PVP scheme, a complex multiclass problem can be decomposed into many linearly separable problems, each of which has only two different training samples. The merit of this task decomposition is that the learning convergence can be guaranteed by using linear discriminant function or GZC discriminant function, and incremental learning can be easily implemented. However, the demerit is that there are a large number of redundant modules in the M^3 network. So how to prune these redundant modules is one of the most important issues for further applying M^3 network to solving large-scale real-world problems. In this paper, we propose a pruning algorithm called *back searching* (BS) algorithm to reduce the redundancy of the M^3 network.

The motivation of the BS algorithm comes from the fact that the actual output of a redundant module must have the same value as the output of the MIN unit when a training sample is presented to the network as an input. Therefore, by using a back-searching technic, the redundant modules can be found out. In the next section we firstly define the redundant problem of the M^3 network. Then we describe our BS algorithm in section III. At last, we present the experimental results and conclusions in section IV and section V, respectively.

II. Redundant problem of the M^3 network

A. Task Decomposition

Let $\mathcal{T}$ be the training set of a K-class classification problem and the K classes are represented by C_1, C_2,..., C_K, respectively.

$$\mathcal{T} = \{(X_l, Y_l)\}_{l=1}^{L} \tag{1}$$

where $X_l \in \mathbf{R^d}$ is the input vector, $Y_l \in \mathbf{R^K}$ is the desired output, and L is the number of training data.

Suppose the K training input sets, $\mathcal{X}_1$, $\mathcal{X}_2$,...,$\mathcal{X}_K$, are expressed as

$$\mathcal{X}_i = \left\{(X_l^{(i)}\right\}_{l=1}^{L_i} \quad \text{for} \quad i = 1, ..., K \tag{2}$$

where L_i is the number of training inputs in class C_i, $X_l^{(i)}$ is the lth sample belonging to class C_i and all of $X_l^{(i)} \in \mathcal{X}_i$ have the same desired outputs, and $\sum_{i=1}^{K} L_i = L$.

It is known that a K-class problem defined by (1) can be divided into $K(K-1)/2$ two-class subproblems, each of which is given by

$$\mathcal{T}_{ij} = \left\{(X_l^{(i)}, +1)\right\}_{l=1}^{L_i} \cup \left\{(X_l^{(j)}, -1)\right\}_{l=1}^{L_j}$$
$$\text{for} \quad i = 1, ..., K-1 \quad \text{and} \quad j = i+1, ..., K \tag{3}$$

Even though a K-class problem is broken into $K(K-1)/2$ relatively smaller two-class problems, some of the two-class problem may be still hard to be learned. We have suggested that $\mathcal{T}_{ij}$ defined by (3) can be further decomposed into a number of two-class subproblems as small as needed according to the class relations among training data [2].

Assume that the input set $\mathcal{X}_i$ defined by (2) is further partitioned into $N_i (1 \le N_i \le L_i)$ subsets in the form of

$$\mathcal{X}_{ij} = \left\{(X_l^{(ij)}\right\}_{l=1}^{L_i^{(j)}} \quad \text{for} \quad j = 1, ..., N_i \tag{4}$$

where $L_i^{(j)}$ is the number of training inputs included in $\mathcal{X}_{ij}$, and $\cup_{j=1}^{N_i} \mathcal{X}_{ij} = \mathcal{X}_i$

After dividing the training input set $\mathcal{X}_i$ into N_i subsets $\mathcal{X}_{ij}$ (4), the training set for each of the smaller and simpler two-class problems can be given by

$$\mathcal{T}_{ij}^{(u,v)} = \left\{(X_l^{(iu)}, +1)\right\}_{l=1}^{L_i^{(u)}} \cup \left\{(X_l^{(jv)}, -1)\right\}_{l=1}^{L_j^{(v)}}$$
$$\text{for} \quad u = 1, ..., N_i, v = 1, ..., N_j$$
$$i = 1, ..., K-1 \quad \text{and} \quad j = i+1, ..., K \tag{5}$$

where $X_l^{(iu)} \in \mathcal{X}_{iu}$ and $X_l^{(jv)} \in \mathcal{X}_{jv}$ are the input vectors belonging to class C_i and class C_j, respectively, $\sum_{u=1}^{N_i} L_i^{(u)} = L_i$, and $\sum_{v=1}^{N_j} L_j^{(v)} = L_j$.

From (5), we see that if $N_i = L_i$, i.e., each of the two-class subprolems contains only two different training data. Obviously, these two-class subproblems are linearly separable problems.

B. Module combination

From (4) and (5), we can see that a K-class problem is divided into

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j \tag{6}$$

two-class subproblems.

If $N_i = L_i$, a K-class classification problem can be divided into many linearly separable subproblems. The number of these linearly separable subproblems can be expressed as

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} L_i \times L_j \tag{7}$$

Obviously, equation (7) shows an upper bound on the number of subproblems that can be obtained by dividing a K-class classification problem into many linearly separable subproblems.

After constructing each individual linear discriminant function as a base classifier, all the individual modules are integrated into a min-max modular network with MIN, MAX, or/and INV units according to two module combination rules [2]. This kind of M^3 network will be referred to as *linear*-M^3 network throughout this paper. A linear-M^3 network is illustrated in Fig. 1.

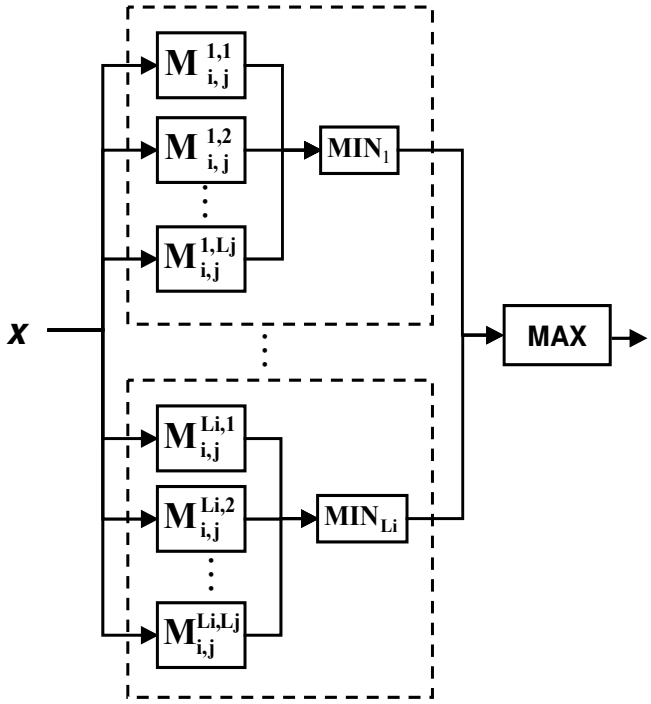

Fig. 1. The M^3 network consists of $L_i \times L_j$ individual network modules, L_i MIN units, and one MAX unit.

C. Reduntant problem of M^3 network

Although we can decompose a complex problem into a series of independent linear separable problems, and then combine them to get the solutions to the original problem, there are a large number of redundant modules in the network when finally integrating all of the trained modules into a M^3 network. For example, see Fig. 2 (a), there are total six lines and each line stands for a linear discriminant function that has been generated by one positive training sample and one

negative training sample. We can see that, in this figure, the boundary of the two different areas (gray and white) is decided only by four of the six lines, while the other two lines are of no use and they are so called redundant modules in the M^3 network. This redundant problem may be a serious problem when the number of linearly separable problems becomes very large, especially for real-world problems. In this paper, we focus on the problem of how to prune the redundant modules in the linear-M^3 network.

D. Terminology definition

Suppose $\mathbf{X}$ is the input space of a M^3 network, U_i is the ith individual network module of a MIN unit, $\{\mathrm{MIN}/U_i\}$ is the residual part of the MIN unit after pruning U_i from this MIN unit, and $\mathbf{T}$ is the training set of the original MIN unit. For the purpose of convenience, we present the definitions of several terminologies as follows.

Definition 1 : For any $x \in \mathbf{X}$, if $U_i(x) \geq \{\mathrm{MIN}/U_i\}(x)$, then we call U_i a ***redundant module*** of the MIN unit.

Definition 2 : If there exists an $x \in \mathbf{X}$, that $U_i(x) < \{\mathrm{MIN}/U_i\}(x)$, then we call U_i a ***non-redundant module*** of the MIN unit.

Definition 3 : For any $x \in \mathbf{T}$, if $U_i(x) \geq \{\mathrm{MIN}/U_i\}(x)$, then we call U_i an ***allowed redundant module*** of the MIN unit.

Definition 4 : The ratio between the number of the pruned modules and the number of the total original modules is defined as the ***degree of pruning***:

$$d_p = \frac{\text{number of pruned modules}}{\text{number of original modules}} \times 100\%$$

Here we have made the definition of the *allowed redundant module* from the view of training set , but not from the view of input space, considering that it is allowed for this kind of modules in the phase of training a classifier. Furthermore, since the MIN and MAX units are similar to the logical AND and OR gates respectively [3], all the redundant modules of each MIN unit will compose of the redundant modules of one MAX module, i.e. the redundant modules of the M^3 network.

III. Back-searching pruning algorithm

In this section we describe our back-searching pruning algorithm for linear-M^3 network. The reason why linear-M^3 network was chosen as the object is that the linear-M^3 network is the relatively simple and basic network among our various kinds of M^3 networks. We can further modify the algorithm presented here for other kind of M^3 networks in a similar way. We only focus on pruning redundant modules of linear-M^3 network throughout this paper.

A. Algorithm

In the BS pruning algorithm, we assume that all of the individual modules in a MIN unit are redundant units firstly, so we tag them as 'false', then by using a 'back-searching' method, we find out all the non-redundant modules in the MIN unit, and at last we delete all the residual modules that are not found out in the end of the algorithm. For a trained linear-M^3 network, we use the following BS algorithm to prune its redundant modules.

1) Suppose a linear-M^3 network is M^3 and has L_i and L_j training samples of class C_i and class C_j respectively. According to two module combination rules, M^3 will include L_i MIN units and one MAX unit, and each MIN units include L_j individual network modules.
2) For each unit $\mathrm{MIN}_k (k = 1, 2, ..., L_i)$ do the following steps:
   a) Tag 'false' to each individual network module $U_r (r = 1, 2, ..., L_j)$ of unit MIN_k.
   b) Suppose the training set of unit MIN_k is $\mathbf{T}_k = \{a_k, b_1, b_2, ..., b_{L_j}\}$, here $a_k \in C_i$ and $b_1, b_2, ..., b_{L_j} \in C_j$.
   c) For each training sample $x \in \mathbf{T}_k$ of unit MIN_k, do the following steps:
      i) Calculate the output value of unit MIN_k, y,
      ii) Search back for all individual network modules whose outputs have the same value with y,
      iii) If $U_r(x) < \{\mathrm{MIN}_k/U_r\}(x)$, then U_r module be tagged as 'true'.
   d) Delete all individual network modules that are marked with 'false'.
3) End.

B. Propositions

In this subsection, we prove the sufficient and necessary propositions for the BS algorithm theoretically.

Proposition 1: (A sufficient proposition) All the redundant modules of a trained M^3 network are pruned by the BS algorithm.

Proof: Suppose U_r is one of the redundant modules of M^3, $\mathbf{T}$ is the training set of U_r, and $\mathbf{X}$ is the input space of M^3, then we have $\mathbf{T} \subset \mathbf{X}$.

From the definition of redundant modules (see Definition 1) and $\mathbf{T} \subset \mathbf{X}$, we have $U_r(x) \geq \{\mathrm{MIN}/U_r\}(x)$ for any training sample $x \in \mathbf{T}$, so after running the BS algorithm, U_r will be tagged as 'false' and then be pruned. □

Proposition 2: (A necessary proposition) All modules that have been pruned by BS algorithm must be allowed redundant modules.

Proof: Suppose U_r is one of the individual modules that have been pruned by the BS algorithm, and $\mathbf{T}$ is the training set of U_r. From the BS algorithm, we can conclude that for any training sample $x \in \mathbf{T}$, there is $U_r(x) \geq \{\mathrm{MIN}/U_r\}(x)$, so from the definition of allowed redundant module (see Definition 3), we can see that U_r is an allowed redundant module.□

Suppose a M^3 network has N MIN units and each MIN unit has M training samples, then the time complexity of the BS algorithm is $O(NM^2)$. For a large scale problem, this pruning algorithm may be a time consumed process.

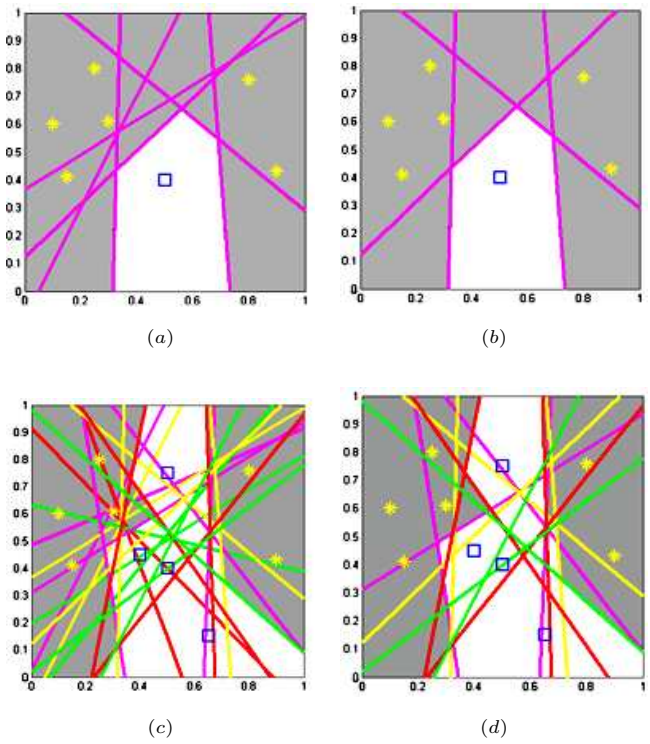

Fig. 2. Illustrations of decision boundaries formed by original linear-M^3 network and pruned linear-M^3 network. (a) The decision boundary formed by a MIN unit and it's individual modules (lines) before pruning; (b) The decision boundary formed by a MIN unit and its individual modules (lines) after pruning; (c) The decison boundary formed by a M^3 network and it's individual modules (lines) before pruning; (d) The decision boundary formed by a M^3 network and it's individual modules (lines) after pruning. Here symbol '□' means a positive data from class $\mathcal{C}_i$, and symbol '*' means a negative data from class $\mathcal{C}_j$).

Fortunately, the whole pruning process can be done before employing the pruned network to real applications. We show some experimental results about time of pruning in Table I.

C. Illustration

To illustrate the BS pruning algorithm, we present a simple example depicted in Fig. 2. In this figure, symbol '□' means a positive sample from class $\mathcal{C}_i$, symbol '*' means a negative sample from class $\mathcal{C}_j$, and each line means a linear discriminant function as well as an individual module of the M^3 network. Each of the individual modules is determined by one positive sample from class $\mathcal{C}_i$ and one negative sample frome the class $\mathcal{C}_j$. From Fig. 2(a), we see that two redundant modules in a MIN unit are pruned by the BS algorithm and the pruning result is depicted in Fig. 2 (b). The boundaries formed by the original M^3 network and the pruned M^3 network are showed in Figs. 2 (c) and (d), respectively. Here 9 redundant modules have been pruned, and the degree of pruning is 37.5%. From Figs. 2 (c) and (d), we can see that the decision boundaries (dark area) before and after pruning are identical.

IV. Experiments

To evaluate the effectiveness of the BS pruning algorithm, we perform six experiments on both benchamark [3] and real-world problems [7]. All the experiments were performed on a 2.8 GHz P4 PC/Win2000.

A. The Two-spirals Problem

In the first experiment, we evaluate the BS pruning algorithm on a toy two-spirals benchmark problem. The data include 194 training samples and 1,700 test samples, respectively. Fig. 3 (a) shows the original two-spirals problem. The blue '□' symbols stand for positive training data and the white '*' symbols stand for negative training data. We use these data to traine a linear-M^3 network and the decision boundary formed by the trained linear-M^3 network is depicted in Fig. 3 (b). The original network consists of 18,432 individual modules. By using the BS pruning algorithm, we prune 15,763 redundant modules from the trained network, and a 85% of degree of pruning is obtained. The decision boundary formed by the pruned network is depicted in Fig. 3 (c). From Figs. 3 (b) and (c), we can see that the decision boundaries before and after pruning are identical. This indicates that the pruned network maintains its generalization performance, while the individual modules and response time are greatly reduced. There are only 2,769 individual modules remained in the pruned network, and the response time is speeded up to 7.33 times in comparison with the original network (see Table I).

B. Iris Plants Database

This data set contains three classes of 50 instances each, where each class refers to a type of iris plant with four data attributes. One class is linearly separable from the other two; the latter are not linearly separable from each other. The first 25 data are used as training data and the remaining data are used as test data in our experiment. There are 3,750 individual modules in the original M^3 network, comparing to 614 individual modules in the pruned M^3 network, a 83% of degree of pruning is obtained and the response time is speeded up to 7.5 times in comparison with the original network. (see Table I).

C. Image Segmentation Data

The image segmentation problem [3] is a real-world problem. The instances were drawn randomly from a database of seven outdoor images. The images were hand-segmented to create a classification for every pixel as one of brick-face, sky, foliage, cement, window, path, and grass. The problem consists of 210 training data and 2,100 test data. The number of attributes is 18 and the number of classes is seven. There are 37,800 individual modules in the original M^3 network, comparing to 8,812 individual modules remained in the pruned M^3 network, a 76% of degree of pruning is obtained and the response time is speeded up to 4.60 times in comparison with the original network (see Table I).

TABLE I

PERFORMANCE COMPARISON OF THE ORIGINAL NETWORK AND THE PRUNED NETWORK ON BENCHMARK AND REAL-WORLD PROBLEMS. HERE TIME UNIT IS MS

Name	Before pruning			After pruning			Time	Degree	Speed up
	# Module	Response time	Correct rate	# Module	Response time	Correct rate	of Pruning	of pruning	
two-spirals	18,432	220	100%	2,769	30	100%	521	85%	7.33
iris	3,750	15	94.67%	614	2	94.67%	10	83%	7.50
image	37,800	9764	87.67%	8,812	2123	87.62%	360	76%	4.60
optdigits	19,691,545	1,637,153	98.00%	16,679,524	14,57,175	98.00%	8,765,985	15%	1.12
Faulty Image1	3,220	2,083	100%	2,352	1,762	100%	144,748	26%	1.82
Faulty Image2	48,000	42,581	97.56%	20,060	22,722	97.56%	5,228,344	58%	1.87

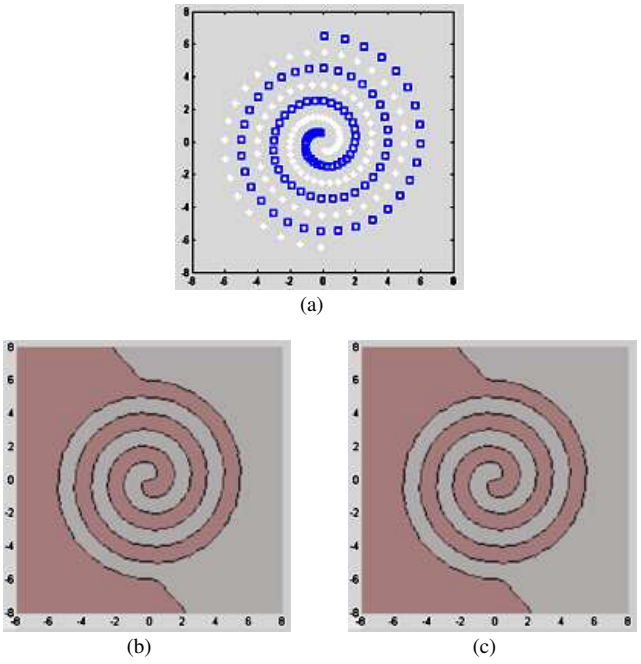

Fig. 3. Two-spirals problem and decision boundary comparison. (a) The training samples of the two-spirals problem; (b) The decision boundary formed by the original M^3 network; (c) The decision boundary formed by the pruned M^3 network.

D. Optical Recognition of Handwritten Digits

The optical recognition of handwritten digits problem [3] is a ten-class classification problem. The instances were drawn from a total of 43 people, 30 contributed to the training set and different 13 to the test set. The problem consists of 3,823 training data and 1,797 test data. The number of attributes is 64 and all the input attributes are integers in the range 0 to16. There are 19,691,545 individual modules in the original M^3 network, comparing to the 16,679,524 individual modules remained in the pruned M^3 network, a 15% of degree of pruning is obtained and the response time of the pruned network is speeded up to 1.12 times in comparison with the original network (see Table I).

E. Faulty image diagnosis

We compare the performance of the original M^3 network with that of the pruned network on an industrial image database acquired from a product line at a leading manufacturing company [7]. The object is to detect the fault image data from large scale correct image database. We firstly extract the original image into 64×64=4,096 dimension vectors using wavelet transform, then train the original M^3 network using the extracted training data and examine the performance of the trained original M^3 network. We prune the redundant modules in the original network and then do the same testing processes using the same data set.

The number of training and test samples in the first simulation are 70/23 and 26/23, and in the second simulation they are 400/60 and 100/23 (here 'a/b' means 'a' correct samples and 'b' fault samples).

In the first simulation, there are 3,220 individual modules in the original M^3 network, comparing to 2,352 individual modules remained in the pruned M^3 network, a 26% of degree of pruning is obtained and the response time is speeded up to 1.82 times in comparison with the original network (see Table I).

In the second simulation, there are 48,000 individual modules in the original M^3 network, comparing to 20,060 individual modules remained in the pruned M^3 network, a 58% of degree of pruning is obtained and the response time is speeded up to 1.87 times in comparison with the original network (see Table I).

The results from Table I show that the pruned M^3 networks have almost the same correct recognition rates as the original M^3 networks, while the highest degree of pruning is up to 85% and the test time is speeded up to 7.33 times. From the simulation results, we can see that the degree of pruning is not only relative to the number of training data, but also relative to the number of attributes and the number of classes in the problem. A problem with more training samples, less number of attributes and less number of classes will have more redundant individual modules in M^3 network, and consequently the BS pruning results will be more efficient than others on data sets. The consistence of the degrees of pruning and the response time shows the effectiveness of the BS algorithm.

V. Conclusions

An pruning algorithm has been presented in this paper for pruning the redundant modules in the linear-M^3 network. The central idea of this algorithm is to build up a linear-M^3 network firstly, then use the training data to find out the redundant modules by using a 'back-searching' technic. We have proved the necessary and sufficient propositions to ensure the correctness of the proposed pruning algorithm. The simulation results show that a highest degree of pruning 85% can be obtained and the response time is speeded up to 7.33 times at most, meanwhile the pruned network maintains the same generalization performance as that of the original network. The simulation results indicate our pruning algorithm is efficient and effective.

The pruned network's generalization ability and the various versions of BS algorithm should be further exploited.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China via the grants NSFC 60375022 and NSFC 60473040.

References

[1] B. L. Lu and M. Ito, "Task decomposition based on class relations: a modular neural network architecture for pattern classification", In: Mira, J., Moreno-Diaz, R., Cabestany, J.(eds.), Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science, Springer Vol.1240 (1997) 330-339

[2] 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, no. 5, pp. 1244-1256, 1999.

[3] C. J. Merz and P. M. Murphy, "UCI Repository of mechine learning databases," Univ. California, Dept. Inform. Comput. Sci. Irvine, CA, 1996. Available http://www.ics.uci.edu/ mlearn/MLRepository.html .

[4] B. L. Lu and M. Ichikawa, "Emergent On-line Learning in Min-Max Modular Neural Networks," Proc. of IEEE/INNS Int. Conf. on Neural Networks, Washington DC, USA, pp. 2650-2655, 2001.

[5] B. L. Lu, Q. Ma, M. Ichikawa, and H. Isahara, "Efficient part-of-speech tagging with a min-max modular neural network," Applied Intelligence, vol. 19 pp. 65-81, 2003.

[6] B. L. Lu, J. Shin, and M. Ichikawa, "Massively parallel classification of single-trial EEG signals using a min-max modular neural network," IEEE Trans. Biomedical Engineering, vol. 51, no. 3, pp. 551-558, 2004.

[7] B. Huang and B. L. Lu, "Fault diagnosis for industrial images using a min-max modular neural network," Lecture Notes in Computer Science, vol. 3316, pp. 842-847, 2004.

[8] K. Wu, F. Y. Liu, H. Zhao, and B. L. Lu, "Fast text categorization with a min-max modular support vector machine," to appear in Proc. of 2005 IEEE/INNS Int. Joint Conf. on Neural Networks, Montreal, Canada.

[9] B. L. Lu and M. Ichikawa, "Emergent on-line learning with a Gaussian zero-crossing discriminant function", Proc. of IEEE/INNS Int. Joint Conf. on Neural Networks, Honolulu, USA, pp. 1263-1268, 2002