Structure Pruning Strategies for Min-Max Modular Network*

Yang Yang and Baoliang Lu

Department of Computer Science and Engineering, Shanghai Jiao Tong University, 1954 Hua Shan Rd., Shanghai 200030, China
alayman@sjtu.edu.cn, blu@cs.sjtu.edu.cn

Abstract. The min-max modular network has been shown to be an efficient classifier, especially in solving large-scale and complex pattern classification problems. Despite its high modularity and parallelism, it suffers from quadratic complexity in space when a multiple-class problem is decomposed into a number of linearly separable problems. This paper proposes two new pruning methods and an integrated process to reduce the redundancy of the network and optimize the network structure. We show that our methods can prune a lot of redundant modules in comparison with the original structure while maintaining the generalization accuracy.

1 Introduction

The min-max modular (M^3) network is an efficient modular neural network model for pattern classification[1][2], especially for large-scale and complex multi-class problems[3]. It divides a large-scale, complex problem into a series of smaller two-class problems, each of which is solved by an independent module. We can conduct the learning tasks of every module in parallel and integrate them to get a final solution to the original problem according to two module combination principles. These combination principles also successfully guide the emergent incremental learning[4]. However, we need to learn too many modules when the size of the training data is large while subproblems are small. Considering the situation of incremental learning, since the training data are presented to the network continually and more and more modules are built, the classifier will be inefficient to respond to novel inputs.

To improve the response performance of min-max modular network, we consider reducing its redundancy at two phases. First is the recognition phase. In this phase, we can dynamically decide which modules have influence on the final result and need computing. The other is the training phase. We can hold a training process to the network to prune the redundant modules for the training data. The pruned modules are supposed to be redundant for the whole input space.

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

J. Wang, X. Liao, and Z. Yi (Eds.): ISNN 2005, LNCS 3496, pp. 646–651, 2005.

In addition, other two lines of research for training optimization can be considered. The idea of the first line arises from instance reduction techniques[5]. Since the nearest-neighbor(NN) algorithm and its derivatives suffer from high computational costs and storage requirement, the instance filtering and abstraction approaches are developed to get a small and representative prototype set. We can firstly build such a condensed prototype set, and then use it to construct the modules of the M^3 network. This way is instance pruning. The other line of research is structure pruning. Lian and Lu first developed a back searching (BS) algorithm to prune redundant modules[6]. We extend their work and propose an integrated process to gain a larger module reduction rate and maintain classification accuracy.

2 Redundancy Analysis

According to the task decomposition principle of the M^3 network, a K-class problem is divided into $K \times (K-1)/2$ two-class problems, each of which can be further decomposed into a number of subproblems. A M^3 network with MIN and MAX integrating units can solve a two-class subproblem, and each network module learns a subproblem. Let's consider the situation of incremental learning [4] mentioned before. Suppose the training set of each subproblem has only two different elements. Let $\mathcal{T}$ be the total training set, X_l be the input vector, where $1 \le l \le L$, and L is the number of training data. The desired output y is defined by

$$y = \begin{cases} 1-\epsilon, & \text{if } X_l \in \text{class } \mathcal{C}_i \\ \epsilon, & \text{if } X_l \in \text{class } \overline{\mathcal{C}}_i \end{cases} \tag{1}$$

where ϵ is a small positive real number, $\overline{\mathcal{C}}_i$ denotes all the classes except $\mathcal{C}_i$. Accordingly, the training set of a subproblem has the following form:

$$\begin{gathered} \mathcal{T}_{ij}^{(u,v)} = \{(X_l^{(iu)}, 1-\epsilon) \cup (X_l^{(jv)}, \epsilon)\} \\ \text{for } u = 1, \cdots, L_i,\ v = 1, \cdots, L_j,\ i, j = 1, \cdots, K, \text{ and } j \neq i \end{gathered} \tag{2}$$

where L_i and L_j are the numbers of training data belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively. Hence the two-class problem has $L_i \times L_j$ subproblems. We use first minimization rule then maximization rule to construct the network. Fig. 1 shows the network structure. Since $\mathcal{T}_{ij}^{(u,v)}$ has only two instances, it is obviously a linearly separable problem and can be discriminated by a hyperplane. The optimal hyperplane is the perpendicular bisector of the line joining the two instances. In this way, the decision boundary established by the network can be written as a piecewise linear discriminant function:

$$\bigcup_{1 \le u \le L_i} \Big(\bigcap_{1 \le v \le L_j} L_{i,j}^{u,v} \Big) \tag{3}$$

where $L_{i,j}^{u,v}$ is the hyperplane determined by $M_{i,j}^{u,v}$ trained on $\mathcal{T}_{ij}^{(u,v)}$. And it is trivial to prove that the decision boundary is the same as that of the nearest

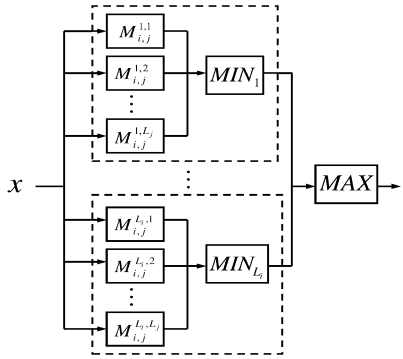

Fig. 1. M^3 network for the two-class problem of $\mathcal{C}_i$ and $\mathcal{C}_j$

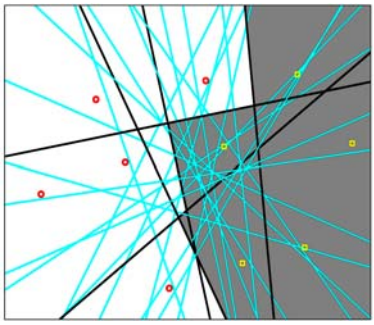

Fig. 2. An example of redundant modules, loops denote $class_1$, and squares denote $class_2$

neighbor classifier. In fact there are a lot of redundancies in the expression above. For example, we can see that only black lines contribute to the boundary, and cyan ones are redundant in Fig. 2, where a 2-D classification problem is depicted.

3 Pruning Algorithms

A backward searching(BS) algorithm for pruning redundant modules has been developed[6], which searches the useful modules based on outputs of every MIN unit. Take $\text{MIN}_k(1 \leq k \leq L_i)$ as an example, set $\mathcal{T}_k = \{a_k, b_1, b_2, \cdots, b_{L_j}\}$, where $a_i \in \mathcal{C}_i$, $b_1, b_2, \cdots, b_{L_j} \in \mathcal{C}_j$. For each instance in $\mathcal{T}_k$, calculate the output of MIN_k and mark the module which gives the minimal value. After all the instances in $\mathcal{T}_k$ are processed, delete those modules without mark. BS algorithm can take off a lot of units while maintaining the original decision boundary. However, it fails to consider removing related redundant MIN units. Here we extend BS algorithm to get an integrated pruning process.

3.1 Our Methods

Because a multi-class problem can be decomposed into a set of two-class problems, here we discuss the redundancy reduction problem of the two-class network for simplicity. Suppose two classes are $class_1$ and $class_2$, which include M and N instances respectively. So the M^3 network contains M MIN units and a MAX unit, each MIN unit contains N modules. We propose two kinds of M^3 Structure Pruning Strategy, called M^3SPS1 and M^3SPS2, respectively. They have different criterions of redundancy.

M^3SPS1. The idea of M^3SPS1 is straightforward, which regards the unit used by no instance in the training set as redundant. Let $\mathcal{T}$ be the training set. The algorithm is as follows:

1. Set Flag(MIN$_i$) = FALSE ($i = 1, 2, \cdots, M$).
2. For each instance I in $\mathcal{T}$:
   (a) Calculate y(I).
   (b) Find MIN$_i$($1 \leq i \leq M$) which has the same value as y(I), and set Flag(MIN$_i$) = TRUE.
3. Delete MIN$_j$($1 \leq j \leq M$) that Flag(MIN$_j$)=FALSE.
4. For each MIN$_i$ retained:

   Execute BS algorithm and prune redundant modules.
5. End.

This approach is consistent with the training data because it just deletes those units which are useless for the classification of the training data.

M^{3}SPS2. The idea of M^{3}SPS2 is similar to Reduced Nearest Neighbor[7] and DROP1[5]. All of them try to conduct reduction without hurting the classification accuracy of the training data set or the subset retained. However, M^{3}SPS2 prune modules other than instances. Moreover, all the instances are guaranteed to be classified correctly no matter whether their related modules have been removed or not. The algorithm is described as follows:

1. Set Flag(MIN$_i$) = FALSE , and create empty list(MIN$_i$), $i = 1, 2, \cdots, M$
2. For each instance I in $\mathcal{T}$:
   (a) Calculate y(I).
   (b) Find MIN$_i$($1 \leq i \leq M$) which has the same value as y(I), set Flag(MIN$_i$) = TRUE, and insert I to list(MIN$_i$).
3. For each MIN$_i$ that Flag(MIN$_i$) = TRUE

   if $\forall\ I \in$ list(MIN$_i$) can be classified correctly without MIN$_i$:

   Set Flag(MIN$_i$) = FALSE.

   For each I $\in$ list(MIN$_i$):

   insert I to list(MIN$_j$), where $1 \leq j \leq M$ and MIN$_j$ gives new y(I) instead of MIN$_i$.
4. Delete MIN$_i$($1 \leq i \leq M$) that Flag(MIN$_i$) = FALSE.
5. For each MIN$_k$($1 \leq k \leq M$) retained:

   Execute BS algorithm and prune redundant modules.
6. End.

This approach is also consistent with the training data. Note that it is sensitive to the presentation order of the MIN units. We can make a search to determine the removing sequence. The basic assumption here is that a module composed by an inner instance is likely to be useless. Since y(I) indicates the perpendicular distance between the instance I and decision boundary, so memorize y($Instance^{(1i)}$) at step 2 and examine the MIN$_i$ in the descending order of y($Instance^{(1i)}$) at step 3 , where $1 \leq i \leq$M, $Instance^{(1i)} \in class_1$ and it is the corresponding instance of MIN$_i$.

4 Experimental Results

We present five experiments to verify our methods, including the two-spirals problem and 4 real world problems. Three of them were conducted on the benchmark data sets from the Machine learning Database Repository[8]: Iris Plants, Image Segmentation and Vehicle Silhouettes. The last one was carried on a data set of glass-board images from an industrial product line, which was used to discriminate the eligible glass-boards. Table 1 shows all the data sets and the numbers of classes, dimensions, training and test samples. Table 2 shows the classification accuracy, response time, and the retention rates of modules and MIN units, where

$$retention\ rate = \frac{\text{No. of modules/MINs after pruning}}{\text{No. of modules/MINs of the original structure}}. \quad (4)$$

Since a multi-class problem can be decomposed into a number of two-class problems, each of which is solved by a M^3 network, for iris, segment and vehicle problems, we record the response time both in parallel and in series. We also list the accuracy and response time of the nearest-neighbor classifier for comparison. All the experiments were performed on a 3GHz Pentium 4 PC with 1GB RAM.

Table 1. Numbers of classes, dimensions, training and test data

	Two-spirals	Iris	Segment	Vehicle	Glass image
Class	2	3	7	4	2
Dimension	2	4	19	18	160
Training	95	135	1540	600	174
Test	96	15	770	246	78

Table 2. Experimental results. For the last three problems, the left sub-column of "time" denotes sequential time and right one denotes parallel time(CPU time, in millisecond)

Problems	1-NN		M^3SPS1					M^3SPS2				
	acc	time	acc	time		MINs	modules	acc	time		MINs	modules
Two-spirals	1.00	31	1.00	32		1.0	0.149	0.99	30		0.432	0.044
Glass image	0.897	47	0.895	63		0.477	0.282	0.885	47		0.075	0.044
Iris	0.982	15	0.982	16	15	0.289	0.042	0.982	15	15	0.067	0.013
Segment	0.958	328	0.956	766	177	0.370	0.036	0.904	453	164	0.053	0.005
Vehicle	0.638	47	0.638	78	41	0.75	0.104	0.630	63	37	0.298	0.047

From the experimental results in Table 2, several observations can be made. M^3SPS1 almost maintains the decision boundary of the original structure, but the pruning ability is limited. M^3SPS2 gains a significant module reduction rate, but the classification accuracy dropped by an average of 1.3%. This is likely due to that the decision boundaries sometimes change dramatically and the classification of the positive points may be affected since too many MIN units have been removed. When handling complex multi-class problems, we can make use of the simple decomposition rules of M^3 and learn the two-class problems in parallel. Then the time can cut down greatly. Take Segment as an example,

parallel M^3SPS2 costs only half the response time as nearest neighbor classifier. And the superiority will be more distinct in solving more complex multi-class problems.

5 Conclusions and Future Work

We have presented two new pruning algorithms for dealing with redundant MIN units and developed a process integrating Back Searching algorithm to reduce the redundancy of the network and optimize the M^3 network structure. The process pays attention to not only massive units but also little modules, and improves the structure significantly. The experiments verified the validity of the structure pruning strategies. Compared with instance reduction methods, they directly concern with decision boundaries, because a module in fact presents a class boundary. So it may express complex concepts more delicately. However, more investigations on the criterion of redundancy are still needed. And it should be noted that this study has examined only hyperplane base classifier. Since the M^3 network is a flexible framework, the user can choose proper base classifier and module size according to different requirements. One future work will investigate methods for pruning the min-max modular networks with other base classifiers, such as the Gaussian zero-crossing function[9], which has an adaptive locally tuned response characteristic.

References

1. Lu, B. L., Ito, M.: Task Decomposition Based on Class Relations: a Modular Neural Network Architecture for Pattern Classification. In: Biological and Artificial Computation: From Neuroscience to Technology. Lecture Notes in Computer Science. Vol. 1240. Springer-Verlag (1997) 330-339
2. Lu, B. L., Ito, M.: Task Decomposition and Module Combination Based on Class Relations: a Modular Neural Network for Pattern Classification. IEEE Trans. Neural Networks, **10** (1999) 1244-1256
3. Lu, B. L., Shin, J., Ichikawa, M.: Massively Parallel Classification of Single-trial EEG Signals Using a Min-Max Modular Neural Network. IEEE Trans. Biomedical Engineering, **51** (2004) 551-558
4. Lu, B. L., Ichikawa, M.: Emergent On-line Learning in Min-max Modular Neural Networks. In: Proc. of IEEE/INNS Int. Conf. on Neural Networks, Washington DC, USA (2001) 2650-2655
5. Wilson, D.R., Martinez, T.R.: Reduction Techniques for Instance-based Learning Algorithms. Machine Learning, **38** (2000) 257-286
6. Lian, H. C., Lu, B. L.: An Algorithm for Pruning Redundant Modules in Min-Max Modular Network (in Chinese). In: Proc. 14th National Conference on Neural Network, Hefei University of Technology Press (2004) 37-42
7. Gates, G. W.: The Reduced Nearest Neighbor Rule. IEEE Transaction on Information Theory, **IT-18-3** (1972) 431-433
8. Blake, C. L., Merz, C. J.: UCI. In: ftp://ftp.ics.uci.edu/pub/machine-learning-databases (1998)
9. Lu, B. L., Ichikawa, M.: Emergent Online Learning with a Gaussian Zero-crossing Discriminant Function. In: Proc. IJCNN'02 (2002) 1263-1268