On Efficient Selection of Binary Classifiers for Min-Max Modular Classifier

Hai Zhao and Bao-Liang Lu
Department of Computer Science and Engineering, Shanghai Jiao Tong University
1954 Hua Shan Rd., Shanghai 200030, China
{zhaohai,blu}@cs.sjtu.edu.cn

Abstract— **Binary classifiers are fundamental components of multiclass pattern classifiers. How to construct a solution to a multiclass problem by efficiently combining the outputs of binary classifiers is a very important issue in neural network and machine learning research. In this paper, we present three different algorithms for selecting binary classifiers for min-max modular classifier to improve its response performance. We also give a theoretical performance estimation of the proposed algorithms. We prove that quadratic complexity of original min-max combination can be reduced to the level of linear complexity in the number of binary classifiers. The experimental results indicate that our proposed algorithms are efficient and effective.**

I. INTRODUCTION

How to construct a solution to a multiclass classification problem by combining the outputs of binary classifiers is one of fundamental issues in neural network and machine learning research. Many popular pattern classification algorithms such as support vector machine (SVM) and AdaBoosting are originally designed for binary classification problems and strongly depend on the technology of multiclass task decomposition and binary classifier combination.

In recent years, this issue had been considered by many researchers in both neural network and machine learning fields. Basically, there are two methods for decomposing multiclass problems. One is one-vs-rest policy, and the other is one-vs-one policy. The former is computationally more expensive, the latter is more popular in practical application and will be concerned in this paper.

There are three main combination policies for one-vs-one scheme according to reported studies. a) *min-max* procedure that comes from one of two stages in min-max modular (M^3) neural network [1], [2]; b) most-win policy or round robin rule (R^3) learning [3]; and c) decision directed acyclic graph (DDAG) [4].

In comparison with one-vs-rest scheme, a shortcoming of one-vs-one decomposition procedure is that it will yield many binary classifier modules, precisely the quantity is the quadratic function of the number of classes, i.e., $K(K-1)/2$. In the recognition phase, however, it is observed that only a part of binary classifiers will be called to produce a solution to the original multiclass problem. In order to improve the response performance of this kind of classifiers, we need to develop efficient algorithms for selecting necessary binary classifiers in the recognition phase. In this paper, we focus on binary classifier selection problem and present three different selection algorithms.

On one hand, in machine learning literature [5], DDAG is superior to other combination policies in many aspects. However, it lacks the capability of recognizing unknown class, which is very important in some applications sometimes and just naturally included in min-max combination. On the another hand, our previous work [6] has shown that min-max combination is much more efficient than most-win policy in an equal and unified optimization procedure.

We start our study in this paper by a distinct way. One difference from related work is that the point we begin is how to optimize the original min-max combination in one-vs-one decomposition scheme, which has not been concerned before. The other difference is that we only care the module based time complexity, which means our work will be independent of the classification algorithms and then it earns more generality. Here we must mention the work on optimization combining policy for multiclass classification [7]. However, this literature focuses in an optimized combining policy for margin based classification, which strongly depends on base classification method, such as AdaBoosting. The work is quite different from our study in this paper, since our work on multiclass combination is completely classification algorithm independent. Naturally, our view on multiclass classification is also different from other work such as [7], [8].

The rest of the paper is organized as follows: In Section II we briefly introduce the min-max combination of binary classifiers. Three different selection algorithms will be presented in Section III. Performance estimation is given in Section IV. The experimental results and discussions on theoretical and experimental results are presented in Section VI and VII. Conclusions of our work and the current line of research are outlined in Section VIII.

II. MIN-MAX COMBINATION

In this section, we give a brief introduction to the min-max combination of binary classifiers. For more details about the whole min-max modular neural network, please refer to [1], [2].

Suppose the training data set for a K-class classification problem is given and the one-vs-one scheme is applied to task decomposition. Thus $K(K-1)$ independent binary classifiers should be assigned in order to solve the corresponding $K(K-$

1) two-class subproblems. K different classes are defined by the set $\{C_0, \cdots, C_{K-1}\}$. A binary classifier that is trained on the data of class C_i and class C_j is denoted by M_{ij}, for $i, j = 0, \cdots, K-1$ and $i \neq j$. For any input sample x, suppose the output of M_{ij} is detonated by $g_{ij}(x)$, then

$$g_{ij}(x) = \begin{cases} 1-\varepsilon, \ x \in C_i \\ \varepsilon, \ x \in C_j \end{cases} \tag{1}$$

where ε is a small real positive number. It is remarkable that M_{ij} may be reused as M_{ji} in combination procedure. Obviously, their outputs are just contrary for the same input sample. Therefore, $K(K-1)$ binary classifiers should be considered in combination, but only a half of them need to be trained, actually. The min-max combination procedure is defined as follows. Let $G(x)$ denote the output vector of the min-max classifier, then

$$G(x) = [G_1(x), G_2(x), \cdots, G_K(x)] \tag{2}$$

where $G_i(x) = \overset{K-1}{\underset{j=0, j\neq i}{\min}} \ g_{ij}(x)$ for $i = 0, \cdots, K-1$.

The min-max classifier is said to assign sample x to class label C_i, if

$$|G_i(x) - (1-\varepsilon)| < \delta \text{ and } |G_j(x) - \varepsilon| < \delta \text{ for } j \neq i \tag{3}$$

where δ is a real number, which denotes the error tolerance. It is obvious that formula (2) and (3) equally define G_i is the maximum of all when class label C_i is assigned to sample x.

For convenient description of the algorithm, three term amendments are given here. a) Class labels are simply defined by the set $\{0, 1, \cdots, K-1\}$. b) The output of a binary classifier will be simply defined by 1 and 0, instead of two intervals in the original description. We say, our selection procedure presented below is only concerned with which one between two classes with respect to the binary classifier, therefore these two label systems are totally equivalent, and so does SVM, which differs two different classes with positive-negative sign. c) We named a set of binary classifier, M_{ij}, for $j = 0, \cdots, K-1$ and $j \neq i$, as a "group", and it is with the group label i. The min-max combination of the outputs of binary classifiers can be redescribed as two steps: Firstly, the minimization operation, *Min*, is applied to all binary classifiers for each group to yield K group outputs. Secondly, the outputs of all the groups are examined by the maximization operation, *Max*. If the result of *Max* procedure is '1', then the label of that group which attribute such result will be the final classification result. Otherwise, the result is an unknown class label. We call the group which leads to the final output class label as " winning group", the others are called as "losing groups".

A min-max combination procedure is illustrated in Fig 1.

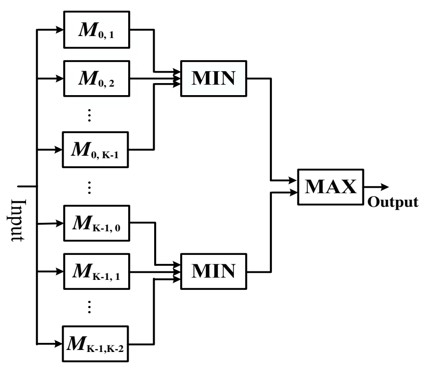

Fig. 1. Illustration of K-class Min-Max combination of $(K-1) \times K$ binary classifiers with K MIN units and one MAX unit

III. Algorithms for Selecting Binary Classifiers

A. Mark Selection Algorithm

The main idea in mark selection algorithm is to fully utilize the output information of checked binary classifier to decide which binary classifier should be selected next.

Mark selection procedure tries to check each group of binary classifiers in turn, and abort the checking operation after a binary classifier without supporting current group label. But it will not certainly check binary classifiers in the next group in turn. Let us consider any checked binary classifier, for example, M_{ij}, which concerned with two classes, class $\mathcal{C}_i$ and class $\mathcal{C}_j$. If the output of M_{ij} holds class $\mathcal{C}_i$, then it means that class $\mathcal{C}_j$ will lose the chance to be a winning class. Therefore, we may give the class label $\mathcal{C}_j$ a failure mark, and the selection algorithm will skip from the group with label j to other group.

The mark selection algorithm can be described as follows.

1) Set the losing tag, $E[i] = 0$ for $i = 0, 1, ..., K-1$.
2) Set $i = 0$.
3) While $i < K$ do
    a) While $E[i] = 1$, do $i = i + 1$.
    b) If $i = K$, then output an unknown class label and the algorithm ends here.
    c) Set the tag of winning group $S = 1$.
    d) For $j = 0, 1, ..., K-1$, do
        i) Present input x to classifier M_{ij} and calculate its output.
        ii) If M_{ij} does not support the current group label i, then set the tag $S = 0$, abort to check the next classifier in the current group, and turn to the next group. Otherwise, set the tag $E[j] = 1$.
    e) If the above operations end normally with the tag $S = 1$, then output class label i, the algorithm ends.
4) If the algorithm does not end at 3rd step and runs here, then output an unknown class label as the final result.

To illustrate the mark selection algorithm, we present an example shown in Table I.

B. Skip Selection Algorithm

Skip selection is a revised version of the mark selection algorithm. The main idea in skip selection is also based on marking the failure group label just like the mark selection algorithm, but it is not turn to the next group while a failure group is found. For example, for the binary classifier M_{ij}, if

TABLE I
ILLUSTRATION OF THE MARK SELECTION ALGORITHM, HERE BOLD VALUES REPRESENT CHECKED BINARY CLASSIFIERS.

	M_{01}=**1**	M_{02}=**0**	M_{03}=0	M_{04}=1	M_{05}=0	losing
M_{10}=0		M_{12}=0	M_{13}=0	M_{14}=0	M_{15}=1	losing
M_{20}=1	M_{21}=**1**		M_{23}=**0**	M_{24}=1	M_{25}=1	losing
M_{30}=1	M_{31}=**1**	M_{32}=1		M_{34}=**1**	M_{35}=**1**	wining
M_{40}=0	M_{41}=1	M_{42}=0	M_{43}=0		M_{45}=0	losing
M_{50}=1	M_{51}=0	M_{52}=0	M_{53}=0	M_{54}=1		losing

the algorithm found it is the first one who does not support the current group label, then the next group to be checked by the algorithm will not with the label $i+1$, but j, instead. Following this operation, the group label to be checked skips into j from i.

The skip selection algorithm can be described as follows.

1) Set losing tag, $E[i] = 0$ for $i = 0, 1, ..., K-1$.
2) Set $i = 0$.
3) Repeat the following operations.
    a) Preassign the tag of winning group found $S = 1$.
    b) While $j = 0, 1, ..., K-1$, do
        i) Present input x to classifier M_{ij} and calculate its output.
        ii) If M_{ij} does not support the current group label i, then
            A) Set the tag $S = 0$.
            B) Set the tag $E[i] = 1$.
            C) Set $i = j$.
            D) If there is $E[j] = 1$, then set $m = 0$, let $m = m+1$ while $E[m] = 1$. Set $i = m$, finally.
            E) If there is still $i = j$ after operations in D), which means any survival group without being excluded can't be found, then output the unknown class label and the algorithm ends here.
        iii) If M_{ij} support the current group label i, then set the tag $E[j] = 1$.
    c) If the above operations are finished normally with $S = 1$, then output class label i as the final result. The algorithm ends.

Table II demonstrates a real skip selection procedure.

TABLE II
ILLUSTRATION OF THE SKIP SELECTION ALGORITHM, HERE BOLD VALUES MEAN THE CHECKED BINARY CLASSIFIERS.

	M_{01}=**1**	M_{02}=**1**	M_{03}=**0**	M_{04}=1	M_{05}=0	losing
M_{10}=0		M_{12}=0	M_{13}=0	M_{14}=0	M_{15}=1	losing
M_{20}=0	M_{21}=1		M_{23}=0	M_{24}=1	M_{25}=1	losing
M_{30}=1	M_{31}=**1**	M_{32}=**1**		M_{34}=**1**	M_{35}=**1**	wining
M_{40}=0	M_{41}=1	M_{42}=0	M_{43}=0		M_{45}=0	losing
M_{50}=1	M_{51}=0	M_{52}=0	M_{53}=0	M_{54}=1		losing

C. Binary Tree Selection Algorithm

The main idea of binary tree selection algorithm for min-max combination totally differs from two selection algorithms mentioned above. It does not check each classifier of each group in turn, instead, it continuously chooses those binary classifiers whose two subscripts are regarded as two group labels, which have not been confirmed as losing groups. Then a removing policy is performed in all groups after checking each chosen binary classifier until a unique winning group label is remained. Such an elimination procedure for each group pair can be described as a binary tree, so we call the selection procedure binary tree selection.

The binary tree selection algorithm can be described in three stages as follows.

The first stage is to initialize input parameters.

1) For $k = 0, 1, ..., K-1$, set the tag $E[k] = 0$.
2) Set the counter which stands for the number of remained survivor group: $T = K$

The second stage is to check each chosen binary classifier to look up for the unique winning group.

1) Repeat the following operations while $T > 1$.
    a) Set $j = 0$
    b) While $j < K-1$, do
        i) While $E[j] = 1$, $j = j+1$.
        ii) If $j = K-1$, then jump to a).
        iii) Choose current j as the first subscript of candidate binary classifier.
        iv) $j = j+1$.
        v) While $E[j] = 1$, $j = j+1$.
        vi) If $j = K-1$, then jump to a).
        vii) Choose current j as the second subscript of candidate binary classifier.
        viii) Check the chosen binary classifier.
        ix) If the output of chosen classifier does not support the group label as one of its two subscripts, then set the corresponding group label with excluded tag.
        x) $T = T-1$.

The third stage is to check the unique remained group to determine whether an unknown result should be output.

1) Find the unique surviving group label by searching the array $E[k]$.
2) Check each classifier in the group. Notice that all classifiers checked in second stage can be omitted.
3) If all checked binary classifiers hold this group label, then output this group label as final combination classification result. Otherwise, any binary classifier's disagreement will lead to an unknown final combination classification result.

Fig. 2 illustrates a real binary tree selection procedure. Here we assume that the states of all outputs of binary classifiers are defined by Fig. 2.

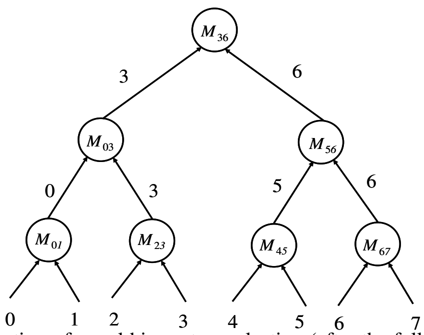

Fig. 2. Illustration of a real binary tree selection (after the following selected checking, another checking for group 3 will be performed to confirm whether it is the winning group.)

IV. Performance Estimation of Selection Algorithms

Now we show that for a K-class problem, the number of checked binary classifiers under binary tree selection algorithm is between $2K-2$ and $K-1$, instead of $K(K-1)/2$ without any selection.

According to the selection procedure, each access for a chosen binary classifier in second stage will cause one group marked as a failure one without any group eliminated duplicately. Then after $K-1$ classifiers are checked $K-1$ group will be certainly marked. Naturally, the unique group is remained. To verify if each classifier in this unique remained group all support the group label, we need to test another $K-1$ classifier at most. So, after $2K-2$ classifiers at most have been checked, we will be sure to obtain the final combination classification result.

For those classification applications those are is not sensitive for unknown property of test samples, we may omit the third stage of binary tree selection algorithm, that is, we simply put the unique remained group label as the final combination classification result in spite of its actual unknown property, which is equally to randomly classify unknown test samples in the original algorithm as some certain class. In fact, such aftereffect can but increases the accuracy of classification. We call the simplified version of binary tree selection algorithm as simple binary tree selection. It is easy to get that only $K-1$ binary classifiers need to check for any test sample under this case.

However, mark and skip selection algorithms, these two selection policies will not be sure to exclude each group label duplicately, which made the selection algorithm fail sometimes. Therefore, these two selection algorithms are not inevitably more efficiently than binary tree selection, though they have many more opportunities to find the wining group ahead than binary tree selection algorithm does, intuitively, which the latter is unfortunately certain to check $K-1$ binary classifiers at least. In fact, according to our experimental results, the upper bound of the number of checked modules in these two selection algorithms is a linear function of K: $aK-b$, where typically, $2 \leq a \leq 3, 1 \leq b \leq 2$.

V. Related Work

DDAG was proposed by Platt [4] for solving multiclass problems with SVMs. For reader's convenience, we redescribe DDAG algorithm as follows.

1) Set all class labels with the tag of success and choose any two different class labels.
2) Checked the binary classifier with the chosen class labels, for example, the binary classifier with labels i and j, namely, M_{ij}.
3) If the chosen binary classifier M_{ij} support the class i, then set class j with the failure tag, otherwise, set i.
4) Continue to choose the class labels from those still successive ones, for example, class j', next checked binary classifier $M_{ij'}$ then can be chosen. Jump into 2. If there is only one class label remained, then continue.
5) Output the survival class label as the final classification result.

Look up for the description of binary tree selection algorithm in section III-C, it is easy to find that DDAG is one revised version of simple binary tree selection algorithm, or vice versa. Simple binary tree selection procedure chooses the unchecked class labels in turn, then it continuously eliminates the failure class through multi-turn choices and testing. While there is not the concept of 'turn' in DDAG combination, DDAG just simply checks any chosen binary classifiers to determine which class label should be eliminated. However, both procedures will eliminate one class from all each time without duplication, which make them equivalent in fact.

The equivalent relationship between DDAG and simple binary tree selection algorithm shows both the close relationship between min-max combination and DDAG and the reason why SVMs under DDAG combination policy takes ideal generalization performance [5].

VI. Experimental Results

Four data sets shown in Table III from Yomiuri News Corpus [9] and UCI Repository [10] have been chosen for this study.

SVMs [11] with the kernel function of RBF are chosen for all binary classifiers, and values of two parameters, C and γ, are given as shown in Table III. Though SVM algorithm is chosen in the experiment, it is not the unique choice for this study. Actually, any classification algorithm can work under our selection schemes. For example, BP neural networks is used traditionally [1], [2].

TABLE III
Distributions of data sets and corresponding parameters for SVMs

Task	#Class	Number of Samples		SVM Parameters	
		Training	Test	C	γ
CoverType	7	348605	232407	128	0.125
Optdigits	10	3823	1797	8	0.0008
Letter	26	15000	5000	4	0.061
Yomiuri	75	913118	181875	64	0.125

The results are shown in Table V. Table V demonstrates the different accuracies under binary tree selection and simple binary tree selection. Experimental results show that our module selection algorithms do speed up the response time of the classifiers. Three module selection algorithms take on similar results. And DDAG and simple binary tree selection give the same results while unknown class label is replaced by a random guess. However, improvement of response time for CoverType data set is not significant. We attribute this to heavily unbalanced scale of each support machine.

TABLE V
COMPARISON OF EXPERIMENTAL RESULTS OF BINARY TREE SELECTION ALGORITHMS ON DIFFERENT DATA SETS

Data set	Correct	Incorrect	Unknown	Average Number of	
	Rate	Rate	Rate	Checked Modules	
(Classes)	(%)	(%)	(%)	Experiment	Bound
CoverType	93.0	6.9	0.2	9.5	12
(7)	93.0	7.0	0.0	6.0	6
Optdigits	98.9	0.8	0.3	15.9	18
(10)	98.9	1.1	0.0	9.0	9
Letter	97.7	2.1	0.1	47.3	50
(26)	97.8	2.2	0.0	25.0	25
Yomiuri	66.4	26.6	7.1	140.3	148
(75)	68.3	31.7	0.0	74.0	74

For three data sets from UCI repository, by removing the last one class of data each time from the obtained subset, we continuously produce corresponding 3-6 class subsets for CoverType data set, 3-25 class subsets for Letter data set and 3-9 class subsets for Optdigits data set. Figures 3 through 5 show experimental results on selection algorithms under similar test set with different number of classes. The number of checked modules in three selection methods take linear function of classes at an experimental view, which agrees on the dissertation at the end of Section IV.

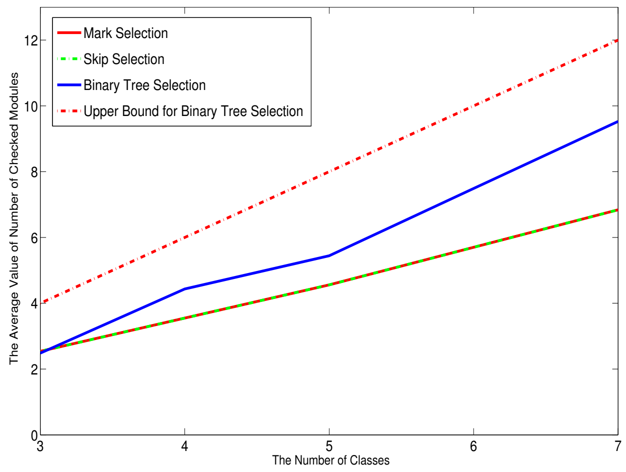

Fig. 3. Comparison of experimental results of CoverType data set, the curves of mark and skip selection algorithms are highly coincident

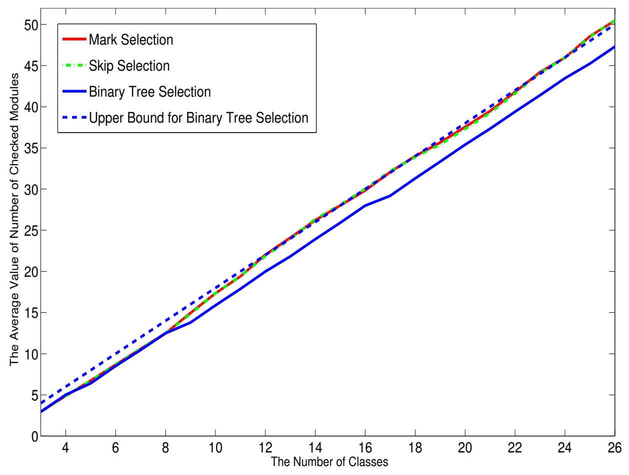

Fig. 4. Comparison of experimental results of Letter data set, the curves of mark and skip selection algorithm are highly coincident

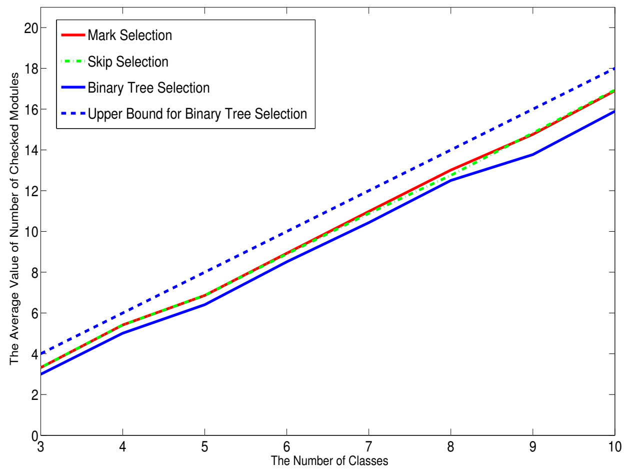

Fig. 5. Comparison of experimental results of Optdigits data set, the curves of mark and skip selection algorithm are highly coincident

VII. DISCUSSIONS

One of our previous work [6] give a complete comparison between most-win policy and min-max policy. We have proved that they are both extreme cases of a unified voting combining policy for one-vs-one decomposition of multiclass problems, where the former takes most high accuracy and the latter takes most high efficiency with a module-based view similarly.

In addition, min-max combination will go into effect in two ways. One is in the situation which unknown label is very necessary, or users want to know how self-confident the classifier is. Typically, text classification with a large number of classes is one kind of such application, where a text often is suitable to be classified into two different classes. Here Fig. 6 shows another case. We restore all images from Optdigits test data set. Fig. 6 (a) shows some typical images for possible digits, and Fig. 6 (b) shows how images of classified unknown digits are. Min-max combination always can give a consistent decision confidence since all its classification results are based on agreement of same number of base classifiers, while most-win or DDAG policy never can do. The other is in theoretical aspect. Min-max modular classifier is a bridge

TABLE IV

COMPARISON OF EXPERIMENTAL RESULTS AMONG SELECTION ALGORITHMS ON DIFFERENT DATA SETS(TIME UNIT OF OPTDIGITS AND LETTER IS 1/1000 SECOND.)

Data set	Average Number of Checked Modules and Response Time									
(classes)	without Selection		Mark Selection		Skip Selection		BT Selection		DAG or SBT Selection	
	#Module	Time (s)	#Module	Time (s)	#Module	Time (s)	#Module	Time (s)	#Module	Time (s)
CoverType(7)	21	4723	6.8	4455	6.8	4480	9.5	4276	6.0	2770
Optdigits(10)	45	3906	16.9	1500	16.9	1531	15.9	1453	9.0	937
Letter(26)	325	123781	50.5	20015	50.5	20031	47.3	19343	25.0	11453
Yomiuri(75)	2775	2470864	99.1	193550	104.9	211750	140.2	247962	74.0	124122

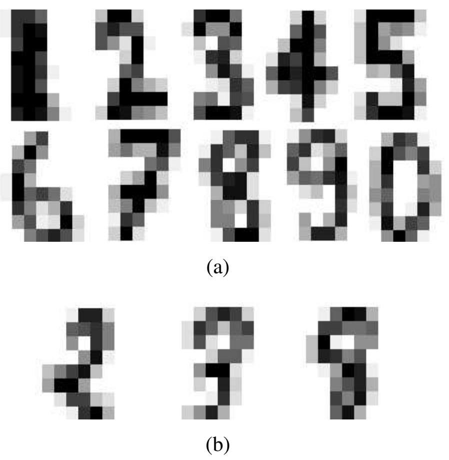
Fig. 6. Image restored from data file of Optdigits test set: typical optical image of digits a) and three images of digits classified as unknown class b)

between most-win policy and DDAG policy. We have shown DDAG can be seen as a partial min-max combination in this paper, and consider the work of comparison between min-max combination policy and most-win combination policy. Therefore we may give a whole view scape for all possible combination under one-vs-one decomposition of multiclass problems. For example, we may be easy to understand why DDAG SVM takes on most merits [5] through the discussions above.

VIII. CONCLUSIONS

In this paper, we start from optimizing min-max combination procedure and finally proved that the number of checked binary classifiers under one-vs-one decomposition of multiclass problems can be reduced to $2K-2$ and $K-1$ for any input. Notice that original min-max combination procedure of binary classifiers with *min-max* rules need to check $K(K-1)/2$ binary classifiers. Our selection procedures significantly improve the response performance of the min-max modular classifiers.

In addition, we built the relationship between two different combination policies, min-max combination and DDAG. we show that DDAG can be regarded as a special case of min-max combination under our selection scheme. Finally, we have presented an analysis method independent of classification algorithms for combing binary classifiers. The method permits us to give a unified performance estimation for any algorithm realization. What's more our analysis has given a briefly theoretical explanation for the reason why min-max combination or DDAG is the most efficient in module combination.

ACKNOWLEDGEMENTS

The authors would like to give their thankfulness to YANG Yang for her excellent programming work and useful advices, LI Jing for proposing a critical amendment advice and WANG Kaian for his generous helpful work on experimental data.

This research was partially supported 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, pp.330-339, 1997.

[2] 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.

[3] J. Frnkranz, "Round Robin Classification", The Journal of Machine Learning Research, Vol.2, pp.721-747, 2002.

[4] J. Platt, N. Cristianini and J. Shawe-Taylor. "Large Margin DAGS for Multiclass Classification", Advances in Neural Information Processing Systems, 12 ed. S.A. Solla, T.K. Leen and K.-R. Muller, MIT Press, 2000.

[5] C.-W. Hsu and C.-J. Lin. "A comparison of methods for multi-class support vector machines", Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

[6] H. Zhao, B. L. Lu, "Combination of Binary Classifiers and Performance Analysis (in Chinese)", Chinese Academic Symposium of Doctoral Student, 2004.

[7] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers", Journal of Machine Learning Research, Vol. 1, pp.113-141, 2000.

[8] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", Journal of Artificial Intelligence Research, Vol. 2, pp.263-286, 1995.

[9] M. Utiyama, H. Isahara, "Large-scale text categorization (in Japanese)", 9th Annual Meeting of the Association (Japan) for Natural Language Processing, pp.385-388, 2003.

[10] C. L. Blake, C. J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[11] B. L. Lu, K. A. Wang, M. Utiyama, H. Isahara, "A part-versus-part method for massively parallel training of support vector machines", Proc. of IEEE/INNS Int. Joint Conf. on Neural Networks (IJCNN2004), pp.735-740, Budabest, Hungary, July 25-29, 2004.