A MULTI-CLASS SVM CLASSIFIER UTILIZING BINARY DECISION TREE

Gjorgji Madzarov, Dejan Gjorgjevikj, Ivica Dimitrovski
Department of Computer Science and Engineering
Faculty of Electrical Engineering and Information Technology
Karpos 2 b.b., 1000 Skopje, Macedonia
Tel: +389 2 3099159; fax: +389 2 3064262
e-mail: madzarovg@feit.ukim.edu.mk

ABSTRACT

In this paper a novel architecture of Support Vector Machine classifiers utilizing binary decision tree (SVM-BDT) for solving multiclass problems is presented. The hierarchy of binary decision subtasks using SVMs is designed with clustering algorithm. For consistency between the clustering model and SVM the clustering model utilizes distance measures at the kernel space, not at the input space. The proposed SVM based Binary Decision Tree architecture takes advantage of both the efficient computation of the decision tree architecture and the high classification accuracy of SVMs. The performance of the proposed SVM-BDT architecture was measured on a problem of recognition of handwritten digits and letters. The experiments were conducted with samples from MNIST, Pendigit, Optdigit and Statlog databases of segmented digits and letters. The results of the experiments indicate that maintaining comparable accuracy, SVM-BDT is faster to be trained than the other methods. Especially in classification, due to its Log complexity, it is much faster than the widely used multi-class SVM methods like "one-against-one" and "one-against-all" for multiclass problems. The experiments showed that this method becomes more favorable as the number of classes in the recognition problem increases.

1 INTRODUCTION

Recent results in pattern recognition have shown that Support Vector Machine (SVM) classifiers often have superior recognition rates in comparison to other classification methods. However, the SVM as a classifier was originally developed for binary decision problems, and its extension to multi-class problems is not straight-forward. The main idea of SVM is to find an optimum hyperplane that separates two classes. This hyperplane is decided by the support vectors which are obtained by solving a quadratic programming problem.

The popular methods for applying SVMs to multi-class classification problems usually decompose the multi-class problems into several two-class problems that can be addressed directly using several SVMs. Although, approaches that address a multi-class problem as a single "all-together" SVM optimization problem exist [1], they are computationally much more expensive than solving several binary problems.

Adapting the SVM classifier to the multi-class scenario is still an ongoing research topic [2]. The conventional way is to decompose the N-class problem into several two-class problems and construct several binary classifiers whose outputs are then mixed to decide the patterns class.

A variety of techniques for decomposition of the multi-class problem into several binary problems using Support Vector Machines as binary classifiers have been proposed, and several widely used are:

• One-against-all

For the N-class problems (N>2), N 2-class SVM classifiers are constructed [3]. The i^{th} SVM is trained while labeling all the samples in the i^{th} class as positive examples and the rest as negative examples. In the recognition phase, a test example is presented to all N SVMs and labeled according to the maximum output among the N classifiers. The disadvantage of this method is that the number of training samples is too large, so it is difficult to train.

• One-against-one

This algorithm constructs N(N-1)/2 2-class classifiers, using all the binary pair-wise combinations of the N classes. Each classifier is trained using the samples of the first class as positive examples and the samples of the second class as negative examples. To combine these classifiers, it naturally adopts Max Wins Algorithm that finds the resultant class by first voting the classes according to the results of each classifier and then choosing the class that is voted most [4]. The disadvantage of this method is that every test sample has to be presented to large number of classifiers (N(N-1)/2). This results in faster training but slower testing, especially when the number of the classes in the problem is big [5].

• Directed acyclic graph SVM (DAGSVM)

Introduced by Platt [6], the algorithm for training a N(N-1)/2 classifiers is the same as in one-against-one. In the recognition phase, DAGSVM depends on a rooted binary directed acyclic graph to make a decision. When a test sample reaches the leaf node, the final decision is made. A test example is presented only to the N-1 SVMs in

the nodes on the decision path. This results in significantly faster testing while keeping very similar recognition rate as One-against-one.

• Binary Tree of SVM (BTS)

This method uses multiple SVMs arranged in a binary tree structure [7]. A SVM in each node of the tree is trained using two of the classes. The algorithm then employs probabilistic outputs to measure the similarity between the remaining samples and the two classes used for training. All samples in the node are assigned to the two subnodes derived from the previously selected classes by similarity. This step repeats on every node until each node contains only one class samples. The main problem that should be considered seriously here is training time, because, one has to test all samples in every node to find out which classes should be assigned to which subnode while building the tree. This may decrease the training performance considerably for huge training datasets.

In this paper we propose a binary decision tree architecture that uses SVMs for making the binary decisions in the nodes. The proposed classifier architecture SVM-BDT (Support Vector Machines classifier utilizing Binary Decision Tree), takes advantage of both the efficient computation of the tree architecture and the high classification accuracy of SVMs. Utilizing this architecture, N-1 SVMs need to be trained for an N-class problem (like in one-against-all), but on the average only $\lceil \log_2 N \rceil$ SVMs are required to be consulted to classify a sample. This can lead to a dramatic improvement in recognition speed when addressing problems with big number of classes.

The Kernel-based clustering introduced to convert the multi-class problems into decision tree problem with SVM classifiers in the nodes of the binary decision tree is explained in Section 2. The results of the experiments of recognition of hanwriteen digits and letters using different multi-class SVM aprroaches are presented in Section 3. Section 4 gives a conclusion of the paper.

2 A MULTI-CLASS SVM CLASSIFIER UTILIZING BINARY DECISION TREE

As shown on (Figure 1), the SVM-BDT solves an N-class pattern recognition problem utilizing a binary decision tree, in which each node makes binary decision using a SVM. The hierarchy of binary decision subtasks should be carefully designed before the training of each binary SVM classifier.

The recognition of each pattern starts at the root of the tree. At each node of the binary tree a decision is being made about the assignment of the input pattern into one of the two possible groups represented by transferring the pattern to the left or right sub-tree. Each of these groups may contain multiple classes.

There exist many ways to divide N classes into 2 groups, and it is critical to have proper grouping for the good performance of SVM-BDT.

For consistency between the clustering model and the way SVM calculates the decision hyperplane the clustering model utilizes distance measures at the kernel space, not at the input space. Because of this, all training samples are modified with the same kernel function that is to be used in the training phase.

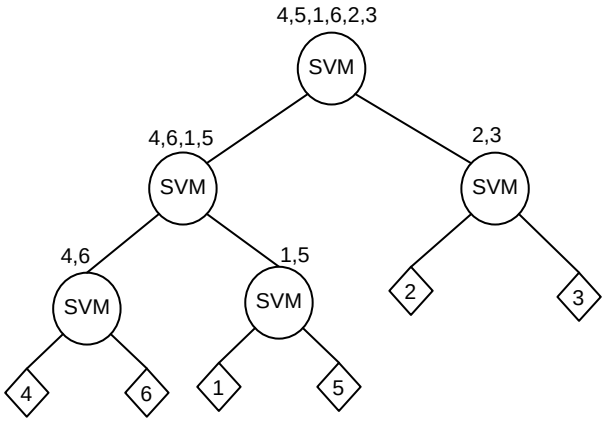

Figure 1: *Illustration of SVM-BDT.*

The SVM-BDT method that we propose is based on recursively dividing the classes in two disjoint groups in every node of the decision tree and training a SVM that will decide in which of the groups the incoming unknown sample should be assigned. The groups are determined by a clustering algorithm according to their class membership.

Let's take a set of samples $x_1, x_2, ..., x_n$ labeled each one by $y_i \in \{c_1, c_2, ..., c_k\}$ where k is the number of classes. The first step of SVM-BDT method starts with dividing the classes in two disjoint groups g_1 and g_2. This is performed by calculating k gravity centers for the k different classes. Then, the two classes that has the biggest Euclidean distance from each other are assigned to each of the two clustering groups. After this, the class with the smallest Euclidean distance from one of the clustering groups is found and assigned to the corresponding group. The gravity center of this group is then recalculated to represent the addition of the samples of the new class to the group. The process continues by finding the next unassigned class that is closest to either of the clustering groups, assigning it to the corresponding group and updating the group's gravity center, until all classes are assigned to one of the two possible groups. This defines a grouping of all the classes in two disjoint groups of classes that are used to train a SVM classifier in the root node of the decision tree. The classes from the first clustering group are being assigned to the first subtree, while the classes of the second clustering group are being assigned to the second subtree. The process continues recursively (dividing each of the groups into two subgroups following the same procedure as explained above), until there is only one class per group which defines a leaf in the decision tree.

For example, on Figure 1 which illustrates clustering of 6 classes, the SVM classifier in the root is trained by considering samples from the classes $\{c_1, c_4, c_5, c_6\}$ as positives examples and samples from the classes $\{c_2, c_3\}$ as negative examples. The SVM classifier in the left child of the root is then trained by considering samples from the classes $\{c_4, c_6\}$ as positives and samples from the classes $\{c_1, c_5\}$ as negative examples. This way, we will train (k−1) SVMs for k-class problem.

3 EXPERIMENTS

In this section, we present the results of our experiments with several multi-class problems. The performance was measured on the problem of recognition of handwritten digits and letters.

Training and testing of the SVMs was performed using a custom developed application that uses the Torch library [8]. For solving the partial binary classification problems SVMs using Gaussian kernel were used.

Here, we compare the results of the proposed SVM-BDT method with the following methods:

1) one-against-all (OvA);
2) one-against-one (OvO);
3) DAGSVM;
4) BTS;

The most important criterion in evaluating the performance of a classifier is usually its recognition rate, but very often the training and testing time of the classifier are equally important.

In our experiments 4 different multi-class classification problems were addressed by each of the 5 previously mentioned methods. For every method the training and testing time and the recognition performance were recorded.

The first problem was recognition of isolated handwritten digits from the MNIST database. The MNIST database [9] contains grayscale images of isolated handwritten digits. From each digit image, after performing a slant correction, 40 features were extracted. The features are consisted of 10 horizontal, 8 vertical and 22 diagonal projections [10]. The MNIST database contains 60.000 training samples, and 10.000 testing samples.

The second and the third problem are 10 class problems from the UCI Repository [11] of machine learning databases: optdigit and pendigit. Pendigit has 16 features, 7494 training samples, and 3498 testing samples. Optdigit has 64 features, 3823 training samples, and 1797 testing samples.

The fourth problem was recognition of isolated handwritten letters – a 26-class problem from the Statlog collection [12]. Statlog-letter contains 15.000 training samples, and 5.000 testing samples, while each sample is represented by 16 features.

The classifiers were trained using all available training samples for the set and were evaluated by recognizing all the test samples for the corresponding set. All tests were performed on personal computer with Intel Core2Duo processor at 1.86GHz on Windows XP.

Tables 1 to 4 show the results of the experiments using 5 different approaches on each of the 4 data sets. The first column of each of the tables describes the combining method of binary SVM classifiers: one-against-all (OvA), one-against-one (OvO), DAGSVM, BTS and SVM-BDT. In the second column the training parameters σ and C are given. The last three columns present the error-rate, the training time and the testing time for the corresponding method.

The results in table 1 show that for the MNIST database (10 classes, large number of samples) OvA method shows the lowest error rate, but is also slowest to train. The other methods show higher but similar error rates. The DAGSVM method shows fastest training and testing times.

Classifier	σ, C	Error-rate(%)	Time(s)	
			Test	train
OvA	2, 100	1.93	23.56	468.94
OvO	2, 100	2.43	26.89	116.96
DAGSVM	2, 100	2.50	9.46	116.96
BTS	2, 100	2.24	26.89	240.73
SVM-BDT	2, 100	2.45	25.33	304.25

Table 1. Recognition results, training and testing times for the MNIST dataset

Classifier	σ, C	Error-rate(%)	Time(s)	
			Test	Train
OvA	60, 100	1.70	1.75	4.99
OvO	60, 100	1.94	3.63	3.11
DAGSVM	60, 100	1.97	0.55	3.11
BTS	60, 100	1.94	0.57	5.21
SVM-BDT	60, 100	1.94	0.54	1.60

Table 2: *Recognition results, training and testing times for the Pendigit, Uci Repository dataset*

From the results in table 2 and table 3 we can see that methods one-against-one (OvO), DAGSVM, BTS and our method SVM-BDT can reach almost the same accuracy. The method one-against-all (OvA) is more accurate than the other methods, which is apparent in both cases. Among all the methods, SVM-BDT is the fastest one in the training phase. Testing time is comparable in methods DAGSVM, BTS and SVM-BDT and they are noticeably better then testing time of one-against-all (OvA) and one-against-one (OvO) methods. However, if the number of the classes is relatively small, the advantage of SVM-BDT is not that evident.

Classifier	σ, C	Error-rate(%)	Time(s)	
			Test	Train
OvA	25, 100	1.17	1.63	3.94
OvO	25, 100	1.55	1.96	2.02
DAGSVM	25, 100	1.67	0.68	2.02
BTS	25, 100	1.51	0.73	5.65
SVM-BDT	25, 100	1.61	0.70	1.59

Table 3: *Recognition results, training and testing times for the Pendigit, Uci Repository dataset*

Classifier	σ, C	Error-rate(%)	Time(s)	
			test	Train
OvA	1.1, 100	3.20	119.5	554.2
OvO	1.1, 100	4.72	160.5	80.9
DAGSVM	1.1, 100	4.74	12.5	80.9
BTS	1.1, 100	4.70	17.2	387.1
SVM-BDT	1.1, 100	4.54	13.1	63.3

Table 4: *Recognition results, training and testing times for the Statlog dataset*

For the three 10-class problems it can be noticed that OvA approach has the lowest error rate. On the other hand, the time needed to train the 10 classifiers for the OvA approach took about 4 times longer than training the 45 classifiers for the OvO and DAGSVM methods. The DAGSVM method showed to be the fastest in the recognition phase but also produces the biggest error rate.

The third problem was recognition of handwritten letters from the Statlog database [12]. Table 4 presents the results of the experiment for this 26-class problem. Again the OvA method showed the lowest error rate but the longest training time. The OvO, DAGSVM and the BTS method achieved very similar error rates that were about 1.5% higher than the OvA method. The DAGSVM is again fastest in recognition being almost 10 times faster than OvA. The time needed for training of the 26 one-against-all SVMs was almost 7 times longer than the time for training the 325 one-against-one SVMs. The BTS method showed the lowest error rate of the methods that use one-against-one SVMs. The SVM-BDT method showed better recognition rate than the methods using one-against-one SVMs while being only slightly slower in recognition than DAGSVM and the fastest while training.

4 CONCLUSION

A novel architecture of Support Vector Machine classifiers utilizing binary decision tree (SVM-BDT) for solving multiclass problems was presented. The SVM-BDT architecture was designed to provide superior multi-class classification performance utilizing a decision tree architecture that requires much less computation for deciding a class for unknown sample. Clustering algorithm that utilizes distance measures at the kernel space is used to convert the multi-class problem into binary decision tree, in which the binary decisions are made by the SVMs. The results of the experiments show that the speed of training and testing are improved, while keeping comparable or offering better recognition rates than the other SVM multi-class methods. The experiments showed that this method becomes more favorable as the number of classes in the recognition problem increases.

References

[1] J. Weston, C. Watkins. Multi-class support vector machines. *Proceedings of ESANN99.* Brussels, Belgium. 1999.

[2] C. W. HSU, C. J. LIN. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*. Vol. 13, pp. 415–425, 2002.

[3] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[4] J. H. Friedman. Another approach to polychotomous classification. Technical report. Department of Statistics, Stanford University, 1997.

[5] P. Xu, A. K. Chan. Support vector machine for multi-class signal classification with unbalanced samples. *Proceedings of the International Joint Conference on Neural Networks 2003*. Portland, 2003, pp.1116-1119

[6] Platt, N. Cristianini, J. Shawe-Taylor. Large margin DAGSVM's for multiclass classification. *Advances in Neural Information Processing System.* vol. 12, pp. 547–553, 2000.

[7] B. Fei, J. Liu. Binary Tree of SVM: A New Fast Multiclass Training and Classification Algorithm. *IEEE Transaction on neural networks*. vol. 17, no. 3, May 2006.

[8] R. Collobert, S. Bengio, J. Mariéthoz. Torch: a modular machine learning software library. Technical Report IDIAP-RR 02-46, IDIAP, 2002.

[9] __, MNIST, MiniNIST, USA http://yann.lecun.com/exdb/mnist

[10] D. Gorgevik, D. Cakmakov. An Efficient Three-Stage Classifier for Handwritten Digit Recognition. *Proceedings of 17th Int. Conference on Pattern Recognition, ICPR2004*. Vol. 4, pp. 507-510, IEEE Computer Society, Cambridge, UK, 23-26 August 2004.

[11] C. Blake, E. Keogh, and C. Merz, UCI Repository of Machine Learning Databases, (1998). http://www.ics.uci.edu/mlearn/MLRepository.html [Online]

[12] Statlog Data Set, ftp://ftp.ncc.up.pt/pub/statlog/ [Online]