An Improved Directed Acyclic Graph Support Vector Machine

Miao YU, Adel RHUMA, Syed Mohsen NAQVI, Jonathon CHAMBERS

(*Advanced Signal Processing Group, Electronic and Electrical Engineering Department, Loughborough University, Loughborough, Leics LE11 3TU, UK*)

Abstract – In this paper, we propose an improved Directed Acyclic Graph Support Vector Machine (DAGSVM) for multi-class classification. Compared with the traditional DAGSVM, the improved version has advantages that the structure of the directed acyclic graph is not chosen random and fixed, and it can be adaptive to be optimal according to the incoming testing samples, thus it has a good generalization performance. From experiments on six datasets, we can see that the proposed improved version of DAGSVM is better than the traditional one with respect to the accuracy rate.

Key words – *class classification*; *directed acyclic graph*; *support vector machine*

Manuscript Number: 1674-8042(2011)04-0367-04

doi: 10.3969/j.issn.1674-8042.2011.04.014

1 Introduction

Multi-class classification is always an important problem in pattern recognition, and there are various of methods for achieving it. Among these methods, Support Vector Machine (SVM) is one of the most popular one. As proposed by Ref. [1], support vector machine is a new generation learning system based on statistical learning theory, and it has good generalization performance compared with the traditional neural network.

However, traditional SVM only aims to solve the two-class classification problem. And recently there are lots of its modifications to be applied to solve the multi-class classification problem. The representative ones are 1 − v − r[1] and 1 − v − 1[2] approaches; however, the drawbacks for these two methods are: ① there are no generalization boundaries; ② both of them will introduce some undivided regions; ③ the computational time is long thus they are not efficient. In order to solve these problems, in Ref. [3], J. Platt et al. proposed a Directed Acyclic Graph (DAG) scheme. In this method, the number of two-class trained SVMs is the same as 1 − v − 1 version but the decision process is carried out according to a tree-like structure compared with the 1 − v − 1 SVM, so the decision time is shorter. And in Ref. [3], J. Platt et al. have given out proof of the generalization boundary for DAGSVM, so its generalization performance can be guaranteed.

As proposed by Ref. [3], the DDAG is equivalent to operating on a list. For each iteration of the decision process, one class is eliminated from the list. However, we notice that the choice of the class order in the list is arbitrary. For different class order the generalization performance is definitely different, a random choice can not guarantee a good generalization performance. In order to solve this problem, we propose an adaptively structured DAGSVM. This new version of DAGSVM structure is no longer fixed, instead, it is determined by the separation levels of different classes, which are evaluated by distance between centers of two classes in the feature space, and adapts according to the incoming test samples. The structure of this paper is shown as follows: Section 2 briefly introduces the fundamentals of two classes SVM and DAGSVM, in section 3, the new version of DAGSVM is proposed and section 4 gives the comparisons for the traditional DAGSVM and the new version of DAGSVM on six different datasets.

2 Support vector machine and DAGSVM for multi-class classification

2.1 Fundamentals of two-class SVM

For a labeled dataset, $\{x_i, y_i\}$, $i = 1 \cdots 1$, $y_i \in \{-1, 1\}$, $x_i \in \boldsymbol{R}^d$, the aim of SVM is to find a hyper plane which separates the two classes while keeping the "margin" between these two classes maximum, that is, to solve the following problem as shown in Eq. (1).

* **Received:** 2011-06-15
Corresponding author: Miao YU (elmy@lboro.ac.uk)

$$\min_{w,\xi}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i\right\},$$
$$s.t.\quad y_i(w\cdot x_i - b)\geqslant 1-\xi_i,\ \xi_i\geqslant 0. \qquad (1)$$

In Eq. (1), the term ξ_i means slack variable which is introduced to cope with noises in the training dataset, and this form of SVM is known as "soft margin" SVM[1].

Eq. (1) is the primal form of SVM, in order to solve it, an equivalent dual problem is introduced. By introducing Lagrange multipliers and using the Karush-Kuhn-Tucker conditions, the original problem is converted to

$$L(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i\cdot x_j,$$
$$s.t.\quad \alpha_i\geqslant 0,\quad \sum_{i=1}^{n}\alpha_i y_i = 0. \qquad (2)$$

Solution for the original problem is

$$w = \sum_{i=1}^{N_s}\alpha_i y_i x_i, \qquad (3)$$

and the decision function for a two-class SVM is represented as

$$f(x) = \sum_{i=1}^{N}\alpha_i y_i x_i\cdot x + b. \qquad (4)$$

Normally, the original data is mapped into high dimension feature space: $x_i \sim \Phi(x_i)$. And in Eq. (2) and Eq. (4), we can use $\Phi(x_i)\cdot\Phi(x_j)$ to replace $x_i\cdot x_j$. With the concept of kernel function is introduced, we use a kernel function $k(x_i, x_j)$ to represent $\Phi(x_i)\cdot\Phi(x_j)$. The most commonly used kernel function is RBF kernel which is defined as

$$k(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2). \qquad (5)$$

The advantage of RBF kernel is that it can map the original data into an infinite dimensional feature space, which can increase the two classes' separability. We use RBF kernel throughout this paper.

2.2 DAGSVM

The DAGSVM, proposed by Ref. [3], is a new version of multi-class SVM algorithm. Compared with the traditional $1-\mathrm{v}-1$ and $1-\mathrm{v}-\mathrm{a}$, it is efficient and has a good generalization performance. For DAGSVM in a N-class classification problem, $(N-1)*N/2$ two-class SVMs are trained, which is equivalent to $1-v-1$ SVM; however, it has a faster decision process than $1-\mathrm{v}-1$ version, because it only needs to be judged by $(N-1)$ two-class SVMs instead of $(N\text{-}1)*N/2$ for decision process. Fig. 1 shows the decision process for a four-class classification problem.

And the decision process for one node (here we take the root node as an example) is shown in Fig. 2.

The DAGSVM has a tree-like structure. And for an incoming test sample, the decision process just follows the structure and a decision is made when one leaf node is reached.

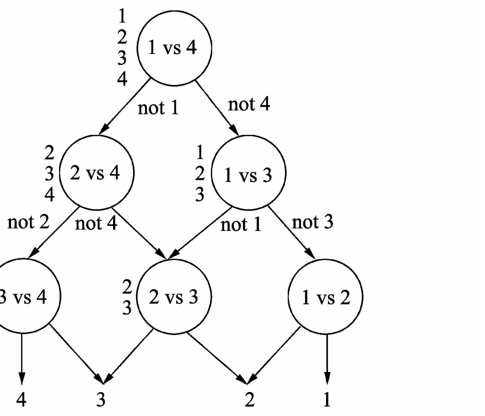

Fig. 1 The decision process for the DAGSVM (this figure is cited from Ref. [3])

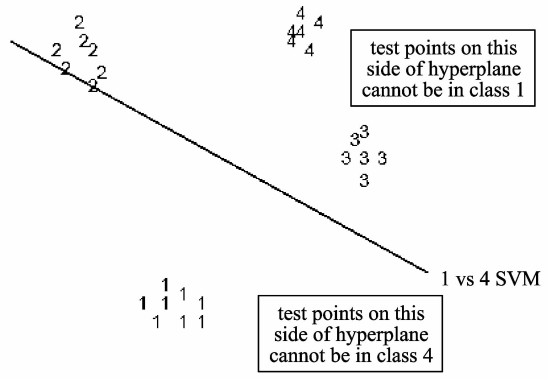
test points on this side of hyperplane cannot be in class 1
1 vs 4 SVM
test points on this side of hyperplane cannot be in class 4

Fig. 2 A diagram of the input space of a four-class problem (A 1 − v − 1 SVM can only exclude one class from consideration (this figure is cited from Ref. [3]))

From Fig. 1, an important fact we can observe is that the decision process for DAGSVM is just like operating on a list, where each node eliminates one class from the list. The list is initialized with a list of all classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the other class is eliminated from the list, and the DDAG proceeds to test the first and last elements of the new list. The DDAG terminates when only one class remains in the list.

3 An improved version of DAGSVM

However, one disadvantage of DAGSVM is that, due to order of the class number in the list is random, a high generalization performance can not be guaranteed. In order to solve this problem, we propose a method which can determine the order automatically according to the incoming test sample.

It is straightforward that the two classes with highest separation level should be put as first and last element of the list (corresponding to the node of the DAGSVM structure). To measure the separation level, we use the distance between two classes (DBTC). The calculation of DBTC in the kernel space is defined in Eq. (6).

$$\begin{aligned}DBTC &= \| m_1^{\varphi} - m_2^{\varphi} \| \\ &= (m_1^{\varphi} - m_2^{\varphi})^{\mathrm{T}}(m_1^{\varphi} - m_2^{\varphi}) \\ &= \frac{1}{n_1^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1} k(x_{1,i}, x_{1,j}) + \\ &\quad \frac{1}{n_2^2}\sum_{i=1}^{n_2}\sum_{j=1}^{n_2} k(x_{2,i}, x_{2,j}) - \\ &\quad \frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k(x_{1,i}, x_{2,j}), \end{aligned} \tag{6}$$

where $k(x, y)$ represents the RBF kernel function, and m_1^{φ} and m_2^{φ} are means of two classes in the feature space defined as

$$m_1^{\varphi} = \frac{1}{n_1}\sum_{i=1}^{n_1} \varphi(x_{1,i}), \tag{7}$$

$$m_2^{\varphi} = \frac{1}{n_2}\sum_{i=1}^{n_2} \varphi(x_{2,i}). \tag{8}$$

Although it seems that the DBTC value only considers inter-distance while ignoring the within-class data distribution; however, by a simple modification, we can find that under the RBF kernel assumption, Eq. (9) is obtained.

$$\begin{aligned}DBTC &= \Big(2 - \frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k(x_{1,i}, x_{2,j})\Big) \\ &\quad - \Big(1 - \frac{1}{n_1^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1} k(x_{1,i}, x_{1,j})\Big) \\ &\quad - \Big(1 - \frac{1}{n_2^2}\sum_{i=1}^{n_2}\sum_{j=1}^{n_2} k(x_{2,i}, x_{2,j})\Big) \\ &= \frac{d(C_1, C_2)}{n_1 n_2} - \frac{d(C_1, C_1)}{n_1^2} \\ &\quad - \frac{d(C_2, C_2)}{n_2^2}, \end{aligned} \tag{9}$$

while $d(C_1, C_2)$ is calculated as

$$d(C_i, C_j) = \sum_{x \in C_i}\sum_{y \in C_j} \| \varphi(x) - \varphi(y) \|^2. \tag{10}$$

So, we can find out that the DBTC value both considers the inter-class distance and the within class distribution from Eq. (9). And as proposed in Ref. [4], the DBTC measure is just equivalent to Fisher Criteria for the RBF kernel case so that it could be used for measuring the two classes' separation.

After finding out the most separable pair of classes, firstly, we feed the test sample into the two-class SVM constructed by these two classes, and one class will be eliminated after decision. Next, we find out the class which is most separable with the remaining class and use the samples from this class and the remaining class to build up a two-class SVM, and make a second decision. The process is repeated until only one class is left. It should be noted that , initially, the separation level between any pair of classes should be computed beforehand.

The procedure is summarized as follows:

1) Computerize the DBTC distances between any pair of classes.

2) Choose the pairs with the largest DBTC value and put them as first and last elements in the list.

3) For the incoming test sample, use the constructed two-class SVM to make a decision, while one class is eliminated and the other remains. We find out the most separable class (except for the eliminated classes) with the remaining class and build up another two-class SVM.

4) Repeat step 3 until one class is left.

In this way, the structure of the DAGSVM is adjusted adaptively according to the incoming test sample. It guarantees that for every iteration, the two classes used to build up the two-class SVM are always the most separable to guarantee good generalization performance.

4 Experimental results

In the experimental part, we use six datasets (iris, wine, glass, vowel, vehicle, ! segment) from UCI[5]. The details of the dataset we use are shown in Tab. 1.

Tab. 1 The properties of the datasets

Datasets	No. of training samples	No. of testing samples	No. of attributes	No. of classes
Iris	96	54	4	3
Wine	107	71	13	3
Glass	128	86	9	6
Vowel	528	462	10	11
Vehicle	470	376	18	4
Segment	1 310	1 000	18	7

For the ith attribute of sample x, it is normalized into [−1, 1] by the following Eq. (11).

$$x_i = 2 \cdot \frac{x_i - min}{max - min} - 1. \tag{11}$$

While *min* and *max* are the minimum and maximum value for the ith attribute obtained from the training dataset, in this way the classification will not be affected by some dominant attributes.

For a SVM, the parameters should be tuned in order to achieve a good performance. For parameter tuning, instead of the time consuming cross validation, we use a method proposed in Ref. [6] to tune the parameters (C and γ) for RBF kernel SVM. It is described as the two following steps:

1) For each kernel parameter γ, we calculate the DBTC value for two classes and choose optimal γ which maximizes the DBTC.

2) For each C, we train the classifier and choose the optimal C which maximizes the n-fold validation accuracy. In our experiment, we choose n to be 10.

After tuning each two-class SVM at the optimal parameter set, we train every pair of two-class SVM and compare performance of the traditional DAGSVM with the improved version proposed in this paper. For the traditional version, the order of the class sequence in the list is chosen randomly for twenty times. And the maximum accuracy, minimum accuracy and average accuracy are presented. The results are shown in Tab.2.

From the experimental results of the six datasets, we can see that accuracy obtained by our method is equal to or higher than the maximum accuracy of the traditional DAGSVM among twenty times' random choice in the order of the class list, thus proving that our method achieves a better generalization performance than the traditional DAGSVM.

Tab.2 Experimental results of six datasets

Datasets	Accuracy of traditional DAGSVM (20 times, the order of the list is chosen randomly each time)(%)			Accuracy of the proposed method (%)
	max	min	Ave.	
Iris	92.59	92.59	92.59	92.59
Wine	95.77	95.77	95.77	95.77
Glass	66.28	63.95	65.93	66.28
Vowel	61.47	61.26	61.46	62.99
Vehicle	77.39	77.13	77.21	77.39
Segment	97	96.8%	96.81	97

5 Conclusions

In this paper, we propose an improved version of DAGSVM. Compared with the traditional version which chooses the order randomly, this improved one can determine the order adaptively according to the incoming test sample and obtain a better generalization performance. It inherits advantage of the traditional DAGSVM that the computational time is faster than the 1-v-1 and 1-v-a while achieving a higher accuracy rate than the traditional DAGSVM.

References

[1] V. Vapnik, 1998. Statistical learning theory. New York: Wiley.
[2] J. Friedman, 1996. Another approach to polychotomous classification. Technical report, Stanford Department of Statistics.
[3] J. Platt, N. Cristianini, J. Shawe-taylor, 2000. Large margin DAGs for multiclass classification. Advances in Neural Information Processing Systems, MIT Press, p.547-553.
[4] J. Sun, C. Zheng, X. Li, et al., 2010. Analysis of the distance between two classes for tuning SVM hyperparameters. *IEEE Trans. on Neurral Networks*, 21(2): 305-318.
[5] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/.
[6] K. Wu, S. Wang, 2009. Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. *Pattern Recognition*, 42: 710-717.