LATTICESVM——A New Method for Multi-class Support Vector Machines

Liu Zhibin, Jin Lianwen[+]

Abstract*—*Multi-class approaches for SVM (Support Vector Machines) is a very important issue for solving many practical problems (such as OCR and face recognition), since SVM was originally designed for binary class classification. Lots of methods based on traditional binary SVM have been proposed, each with its advantages and disadvantages. Among them, one-versus-one, one-versus-all, directed acyclic graph and binary tree are four most widely used methods. In this paper a novel LATTICESVM method, which can significantly reduce the storage and computational complexity, is proposed for multi-class SVM. A comparison in terms of storage, classification speed and accuracy against the four traditional multi-class approaches is given through both theoretic analysis and experiments on large scale handwritten Chinese character recognition. The results obtained clearly show the effectiveness of the proposed method.

I. Introduction

Support Vector Machines (SVM) is a learning machine based on the structural risk minimization induction principle [1], which has achieved superior performance in a wide range of applications, such as OCR, face recognition and so on. The basic concept and theory, which have been described in many previous papers [2][3][4][5], are omitted here. However, Support Vector Machines (SVM) was originally designed for binary classification. The multi-class classification problem is conventionally solved by decomposition of the problem into several two-class problems to which binary SVMs can be applied. Among them, one-versus-one (OVO) SVM [6][7], one-versus-all(OVA) SVM [8], directed acyclic graph (DAG)SVM [9]and binary tree (BT)SVM [10][11] are four typical and the most popular methods. These methods require either to construct a large number of binary classifiers or to solve a larger-scale optimization problem, which are computationally expensive

In this paper, we propose a novel LATTICESVM method, which can successfully solve the multi-class problem and achieve good performance in terms of storage and computational complexity. The theoretical analysis of LATTICESVM is presented and the experiments on Handwritten Chinese Characters Recognition (HCCR) have been performed to validate the proposed algorithm.

Manuscript received November 30, 2007. This paper is partially sponsored by the following research foundations: New Century Excellent Talent Program of MOE of China (No.NCET-05-0736) and NSFC (No. U0735004), GDNSF (07118074)

Liu Zhibin is with the School of Electronics and Information Engineering, South China University of Technology, WuShan Road, Guanzhou City, Guandong Province,510640, China (e-mail: zhibin.liu08@gmail.com)

Jin Lianwen is with the School of Electronics and Information Engineering, South China University of Technology, WuShan Road, Guanzhou City, Guandong Province,510640, China (corresponding author, e-mail: lianwen.jin@gmail.com).

A. Brief Review of Four Typical Multi-class SVM Approaches

1) One-versus-One Strategy (OVO)

This method was first proposed for SVM in [6][7]. In this method, $(N*N-1)/2$ classifiers are constructed and each one is trained on data from two classes. During test, Hsu and Lin [12] used the voting strategy of "Max-Wins" and if two classes have identical votes, the one with the smaller index is selected for the output.

2) One-versus-All Strategy (OVA)

The basic idea of OVA is introduced in [14]. In this method N SVM models are constructed, where N is the number of classes. Here the $i-\text{th}$ classifier is trained on the whole training data set in order to classify the members of class W_i against the rest. Therefore, the training examples have to be relabeled: members of the W_i class labeled to 1; members of the other classes labeled to -1.

In the classification phase, the classifier with the largest output defines the estimated class label.

To achieve final result, the OVO method has to test $(N*N-1)/2$ two-class SVMs, while in the OVA approach only N SVMs is tested. However, each of them is trained on the whole data set. Hence, both of them consume a lot of time in the testing phase, which prevent them from practical applications for large-scale pattern recognition tasks. To achieve better performance, lots of improvements are proposed to achieve better performance. The DAG and the BT methods are two typical ones.

3) Directed Acyclic Graph SVM (DAGSVM)

DAGSVM was first proposed in [9]. Its training phase is the same as the one-versus-one method. However, in the testing phase, the decision is obtained by using a rooted binary directed acyclic graph which has internal nodes and leaves (see Fig1). The leaves represent the individual classes, and the corresponding node is a binary SVM classifier of the $i-\text{th}$ and $\mathrm{j}-\text{th}$ classes. Given a test instance, it starts at the root node and evaluates the binary decision function. Then it moves to either left or right node depending on the corresponding output value of the parent node.

978-1-4244-1821-3/08/$25.00©2008 IEEE

Following a specific path, a leaf node, which indicates the predicted result, is reached.

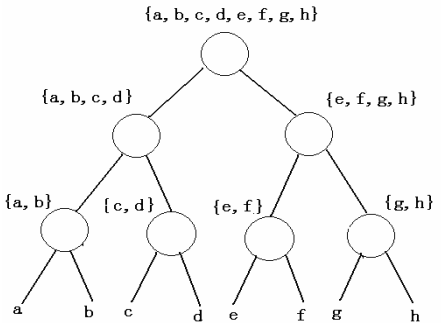

Fig. 1.DAGSVM Classifier

4) Binary tree Strategy (BT)

BT method was based on decision tree, where the multi-class classification problem is decomposed into a series of binary classification sub-problems organized in a hierarchical tree (see Fig2) [11]. The leaves represent the individual classes, and the nodes are classifiers performing binary decision task. Similar to the DAG method, a test instance moves from the parent node to the children node, and finally gets to the leaf that shows the predicted class.

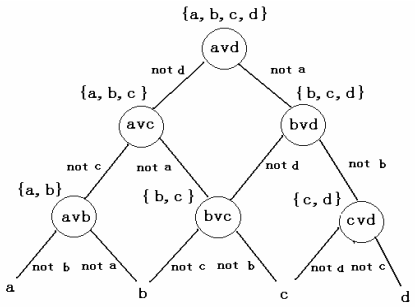

Fig. 2.Binary Tree Classifier

The multi-class methods introduced above have been widely used in many fields, such as digit recognition [8] and face recognition [14].However, the theoretical analyses and the experiments in the later section verify that the computational complexity of these four traditional approaches are proportional to the square of the class number/O(N^2), which prevents them from application in large-scale classification problems such as Chinese characters recognition. To apply SVM on large-scale classification problem, LATTICESVM algorithm for large-scale classification problems is proposed, which successfully reduces the space and computational complexity and shows satisfied performance on recognition rate.

The rest of this paper is organized as follows. The LATTICESVM algorithm is introduced in Section Ⅱ. In Section Ⅲ, a comparative analysis of conventional methods against our new method is presented. Section Ⅳ describes experiments where LATTICESVM applies to HCCR (Handwritten Chinese character recognition) data set. Finally in Section Ⅴ we summarize the paper.

II. LATTICESVM ALGORITHM

Most of the traditional multi-class SVM classification strategies are based on "competition", where different classes compete with each other and the final decision was drawn on all competition results. However, in this paper, we tried to consider the classification problem from a novel aspect of "**Graph**". From our daily life, we noticed that if we want to get to a certain spot, it is not necessary to know the whole map. The only thing we should do is find out the path to the neighbor spot that closer to the destination, and jump from current spot to the next spot (the most famous example is the IP Protocol of the Internet). Inspired by this observation, we proposed a novel classification algorithm that based on graph theory. In this new method, we just need to separate the current class from its neighbor classes, and with these results we can distinguish the current class with the other classes of the feature space more efficiently.

A. Introduction of the algorithm

First, we start with the separable case and then extend the method to the cases that data are not separable.

1) Separable Case

Let x represent the testing sample, and $SVM_{i,j}$ represent the SVM classifier between W_i class and W_j class. If $SVM_{i,j}(x)=i$, we say x belong to W_i class. If $SVM_{i,j}(x)=j$, we say x belong to W_j class.

It is reasonable to assume that in practical problem, the samples of different class locate in different area of the feature space. Therefore, in the ideal case the whole feature space can be split into some small lattices by the boundary of different classes, where each lattice denotes for one class (As shown in Fig3). Supposed that the lattices next to each other are connected by a directed channel, if $SVM_{i,j}(x)=j$, the direction is from W_i to W_j , and if $SVM_{i,j}(x)=i$, the direction is from W_j to W_i.

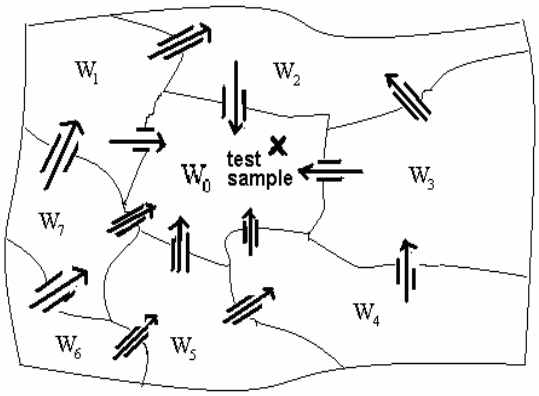

Fig. 3.A Demonstration of Feature Space Split by LATTICE

In consideration of the space limitation, we just list the observations below without proofs.

Observations:

A. If the testing sample belongs to class W_i, then all channels that connected to lattice W_i are directed to W_i.

B. If the testing sample does not belong to class W_i, then at least one channel connected to lattice W_i is directed from W_i to the neighbor lattice.

Based on the Observations, we proposed the multi-class classification algorithm as follow:

LATTICESVM Classification Algorithm:

Step 1. Choose a certain class lattice as the initial lattice randomly and search all the channels connected to the initial lattice.

Step 2. While(there is a channel directed to the neighbor lattice)
{
Move to the neighbor lattice;
Search all the channels connected to the current lattice.
}

Step 3. Classifies the testing sample as the class indicated by the final lattice.

The algorithm we proposed can be regarded as a search method in the feature space. For example, we image that a ball of testing sample is moving through the channels of the lattices in the feature space. It searches all channels connected to the current lattice, and moves to the neighbor lattice once it finds the channel that leaves the current one. Then it searches the new lattice again. According to the observation we list above, the ball will stop moving if and only if it moves into the lattice that it belongs to, therefore the final classification result can be obtained.

2) Nonseparable Cases

If the classes are not completely separable, there may be cycle path through the lattices. As shown in Fig4, $SVM_{i,j}(x) = j$, $SVM_{j,k}(x) = k$, $SVM_{k,i}(x) = i$, the sample ball moves through the lattice circularly. The cycle path means that the sample is so confusing that it is not wise to classify the sample into any class along the cycle path. When such situation occurs, we simply select the one with smaller index. Though it may not be a sound strategy, our experiments show that the results are still satisfying.

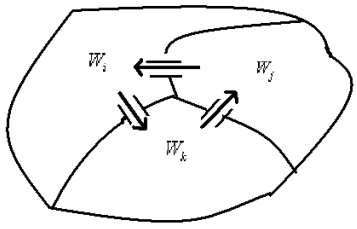

Fig. 4.The Cycle Path in Nonseparable Case

B. Algorithm to Obtain Neighbor Classes

It is obvious that the selection of the neighbor classes is an important issue that influences the performance of the LATTICESVM algorithm. A proper selection of neighbor class can not only improves the rate of prediction but also reduces the computational burden. Inspired by the static-candidate method proposed in [15], we propose several strategies of neighbor selection here and analyze the performance of them in theory. The issue of neighbor selection can be defined as: Given *N* classes and suppose the number of samples per class is M, select K_i neighbor classes for class W_i, where K_i<<*N*.

1) Sample-Sample Distance (SS Method)

It is a natural idea to select the neighbor classes of W_i by considering the distances of every sample of W_i with the samples of other classes. For example, for every sample of W_i, we find out ϕ samples in the feature space that are closest to it. If any of the ϕ samples belongs to another class W_j, we regard W_j as the neighbor of W_i (as shown in Fig5). In this method, all the samples of the feature space are considered abundantly, but at the same time, it brings in great computational complexity. We need to compute distance between each samples of the whole space, so $O(N^2\mathrm{M}^2)$ computational time is required, which results in great computational burden in practice.

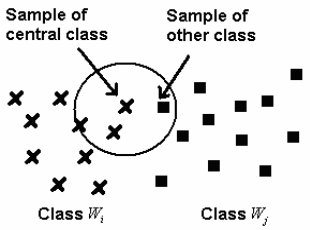

Fig. 5.Sample-Sample Distance Method

2) Center-Center Distance (CC Method)

It is reasonable to assume that the centers of two neighbor classes are always close to each other. Hence, we can first calculate the distances between class W_i and the other classes, and then select K_i classes with the shortest distance to the neighbor classes of W_i. In this strategy, it only needs to calculate all the distances between different class centers, which require $O(N^2)$ computational time. However, it seems not accurate enough to estimate the distribution of the sample only by their class centers. As shown in Fig6, although there is a large distance between the centers of class W_i and W_j, some odd samples of W_j deviate from the general distribution and get close to the W_i.

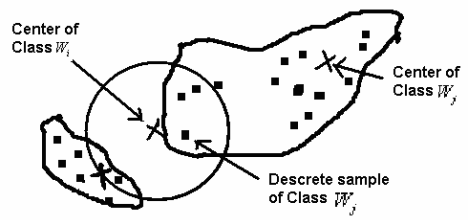

Fig. 6.Odd Points Deviate From the Distribution

3) Sample-Center Distance (SC Method)

The SS and CC methods proposed above consider the sample-sample distance and center-center distance respectively. Now we try to seek the balance and propose another strategy by considering the sample-center distance, with which we can consider the relationship of different class abundantly and release the computational burden at the same time. In this method, we first calculate the distance between each sample of class W_i and the centers of the other classes. Then we figure out the average distances between the samples of the class W_i and the centers of other classes, and select the k classes with smallest distance as the neighbor. This method, which takes $O(N^2\mathrm{M})$ computational time, would be used as the neighbor selection strategy in this paper.

From the discussion above, two more points are worth to be mentioned. First, the distribution of the samples is varied from class to class. Therefore, it is better to select neighbor classes with different numbers according to the situation of each class. Second, the "distance" mentioned above does not only refer to Euclidean Metric but also a metric in general, such as the Manhattan distance and the Mahalanobis distance.

III. Performance analysis of the LATTICESVM against the four traditional multi-class SVMs

In this section, we will briefly analyze the performance of the LATTICESVM in brief and present a comparative discussion for the four traditional multi-class SVM methods mentioned in Section Ⅰ. Here, we assume the number of class is N, and the sample number of each class is M. We also assume that K neighbor classes are selected in LATTICESVM and K is always far smaller than N.

A. Training Time Performance

It is proved in [13] that the training time of binary SVM is decided by the numbers of the training vectors. If applying SMO training [16], the training time [12] is estimated as

$$t_0 = \mu \mathrm{m}^2 \tag{1}$$

Here m represents the number of the training data and μ is a constant. The training time of the multi-class classifier is composed of the training time of each single SVMS involved.

In LATTICE method, the binary SVMs are required between the class W_i and all its neighbor class. So the total computational time is

$$t_1 = KN\mu \mathrm{M}^2/2 = O(NK) \tag{2}$$

For the one-versus-one method $N*(N-1)/2$ two-class SVMS should be trained, therefore the training time is

$$t_2 = N*(N-1)\mu \mathrm{M}^2/2 = O(N^2) \tag{3}$$

For the one-versus-all method, only N SVMs should be trained. However, each SVM is trained on all the samples of the feature space. Hence $\mu(NM)^2$ training time is required for each binary classifier. The training time in total is:

$$t_3 = N\mu(N\mathrm{M})^2 = N^3\mu \mathrm{M}^2 = O(N^3) \tag{4}$$

DAGSVM has the same training time as one-versus-one method so the total computational time is given by

$$t_4 = N*(N-1)\mu \mathrm{M}^2/2 = O(N^2) \tag{5}$$

The training time of binary tree method is influenced by the structure of tree [10]. We construed the tree as shown in Fig2 using the k-means method. The training time of the whole tree is given by

$$t_5 = \sum_{l=1}^{l=\log_2 N} \mu N^2 M^2 / 2^{l-1} \approx 2(N^2 - N)\mu \mathrm{M}^2 = O(N^2) \tag{6}$$

From the fact that K is usually far smaller than N, it can be seen from the above analysis that the LATTICESVM takes the least computational time in the training phase.

B. Testing Time Performance

The testing speed of multi-class SVM is decided by two factors: the total binary tests and the SVs of each binary SVM. The number of SVs is greatly influenced by the distribution of the training data and is assumed to be proportional to the scale of the training data . Therefore, the testing time each binary SVM consumed is

$$T_0 = \alpha m \tag{7}$$

Here α is a constant and m is the number of training data.

In LATTICESVM, the total number of test is influenced by the initial lattice. It is reasonable to estimate that in average $(N-1)/2$ lattices will be traveled through before the final lattice is reached. In each lattice, an expectation of $K/2$ channels should be checked to find out the path to next neighbor. Therefore, the total testing time would be

$$T_1 = K(N-1)\alpha m/4 = O(NK) \tag{8}$$

In the OVO method $N*(N-1)/2$ binary tests are performed, so $O(N^2)$ computational time is required.

The OVA strategy involves N binary SVMs, which are trained on all the training data. Therefore, it takes up $O(N^2)$ time.

In the DAGSVM, each test is mapped to a specific path from the root to certain leaf, so N-1 binary SVM tests are carried out and $O(N)$ testing time is required.

The binary tree method needs to carry out $\log_2 N$ tests, and $O(N)$ computational time is consumed.

As discussed above, the BT and DAG method achieve the best performance in terms of testing speed. However, we noted that the testing time of LATTICE method is greatly influenced by the selection of the initial lattice. In order to reduce the testing time, a two stage strategy is usually applied in practical use. First, the minimum distance classifier is applied to produce a set of candidate classes. And then the candidate classes will be used as the initial lattices for LATTICESVM classifier.

C. Space Performance

The space required by the multi-class SVM is decided by two factors: the total binary SVMs involved and the numbers of support vectors (SVs) of each binary classifier. As the number of SVs is proportional to the scale of the training data, the storage each binary SVM used is assumed as

$$C_0 = \beta m \tag{9}$$

Here β is a constant and m is the number of training data 。

From the analysis above, the LATTICESVM need to construct $KN/2$ binary SVMs. Hence, the total storage required is:

$$C_1 = KN\beta m/2 = O(NK) \tag{10}$$

The analyses of other methods are similar. Both the OVO method and the DAGSVM method need to train $N*(N-1)/2$ binary SVMs, so take $O(N^2)$ storage amount. The one-versus-all method requires building N binary classifiers, which using all the training data. Therefore, $O(N^2)$ storage space is required. In Binary tree method, 2^{l-1} SVMS are trained, hence $O(N\log_2 N)$ storage is needed.

From the discussion above, the LATTICESVM and BTSVM have the best performance in term of storage. And it is worth to mention that it is only a rough estimation here and the storage amount for different method of multi-class SVM is greatly influenced by the distribution of the training data. We noticed that the support vectors always lie on the boundary of two classes. Though both required $O(N^2)$ space, the one-versus-all method takes a much smaller storage burden than the one-versus-one method, as the matter of fact that the boundary in OVA method is much narrower than that in OVO method.

In conclusion, the theoretical analyses of the performance are listed as TABLE I.

TABLE I
THEORETICAL PERFORMANCE COMPARISON OF LATTICESVM AND TRADITIONAL MULTI-CLASS SVM

Strategy	LATTICE	OVO	OVA	DAG	BT
Training time	$O(NK)$	$O(N^2)$	$O(N^3)$	$O(N^2)$	$O(N^2)$
Testing time	$O(NK)$	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N)$
storage	$O(NK)$	$O(N^2)$	$O(N^2)$	$O(N^2)$	$O(N\log_2 N)$

From the table, we can see that the traditional methods show at least $O(\mathrm{N}^2)$ computational complexity in either time or storage consuming, where the proposed LATTICESVM only has $O(NK)$ complexity. From the fact that the neighbor number K is always far smaller than the class number N, we may draw the conclusion that, the LATTICESVM is much efficient for large-scale classification problems.

IV. EXPERIMENT

Handwritten Chinese character recognition (HCCR) problem, which is famous for its huge number of classes and samples, can hardly be solved by the traditional multi-class SVM methods. In order to show the efficiency of the proposed method, we perform the LATTICESVM on large-scale HCCR problem and compare the performance of the proposed method with four traditional multi-class SVM methods introduced in Section I.

The HCL2000 database is used in our experiments, which is collected by Beijing University of Posts and Telecommunications for China 863 project. It contains 3,755 frequently used simplified Chinese characters where all the character images are normalized to 64×64 pixels. A part of experimental samples is illustrated in Fig.7. In these experiments, 512 dimension Gradient features [17] are extracted and Linear Discriminate Analysis (LDA)[18] is performed to reduce the dimension to 256.

Fig.7. Some samples of character "啊" in HCL2000

To reduce the running time and improve the efficiency, we constructed two data sets for the experiments. In the experiments of comparative study between LATTICESVM and traditional multi-class SVM, 500 classes are involved, where 100 samples from each class are used for training and another 30 samples are used for testing. In order to obtain more reliable experimental results, in the pivotal experiment of LATTICESVM for large-scale HCCR problem, we use 300 sets and 200 sets of all the 3755 classes for training and testing respectively. (That means there are total 3755×300 training samples and 3755×200 testing samples)

A. Comparison of LATTICESVM and Traditional Method

In this experiment, we compare the performance of LATTICESVM method and four traditional methods. For binary tree method, the k-means clustering algorithm is applied to construct the binary tree. For LATTICESVM, the SC method that described in section Ⅱ, which takes about two hours here, will be used to select 10 neighbors for each class. And by calculation, 2990 binary SVMs should be built. In the testing phase the two stage strategy introduced in section Ⅲ is applied. In the experiment, a PC with Pentium 3.0G processor and 1G memory is used. It is worth noting that, in these experiments only 500 classes are used for the reason that it is hardly to apply the traditional multi-class methods to too large-scale problems, since the computational complexity of these methods are proportional to the square of the class number.

TABLE Ⅱ
PERFORMANCE COMPARISON OF LATTICESVM AND TRADITIONAL MULTI-CLASS SVM

Strategy	LATTICE	OVO	OVA	DAG	BT
Training Time (hour)	**2.3**	5.3	14.9	5.3	5.5
Testing Time (ms/character)	**65**	90000	1408	400	90
Storage (GB)	**0.15**	7.31	0.23	7.31	0.13
Accuracy (%)	**98.89**	98.93	98.88	98.93	96.50

From Table Ⅱ, we can clearly see that the LATTICESVM outperforms other methods in term of speed in both the training phase and testing phase. The BT method achieves best performance in term of storage and the OVO method is superior to other methods in accuracy. However, our method also achieves similar storage and accuracy performance.

Totally speaking, the OVO method performs best in accuracy, however, the great burden of testing time and storage prevent it from practical application. The DAG method performs as well as the OVO method in accuracy and the testing speed is much faster, but the great storage is still a problem. The OVA method shows satisfied performance both in the storage and accuracy. However it takes more time in both the training and testing phase. The BT strategy shows promising performance on speed and storage, but unluckily, except in accuracy. The LATTICE method, which is superior to other methods in the terms of speed and storage, also shows satisfying capability in accuracy. From the results above, we may draw the conclusion that the LATTICESVM is a valid algorithm for multi-class problems and shows extremely excellent performance particular for large-scale classification problems

B. LATTICESVM for Large Scale HCCR

In order to show the validity of the LATTICESVM for large-scale HCCR problem, we experiment on all the 3,755 classes of HCL2000 database, and among them 300 sets data are used for training and 200 sets for testing. Here we compare the traditional minimum distance classifier plus LDA method with our two-stage method proposed in Section Ⅲ. The result is shown in tableⅢ.

TABLE Ⅲ
PERFORMANCE COMPARISON OF LATTICESVM AND MINIMUM DISTANCE CLASSIFIER

classifier	Minimum distance classifier +LDA	Minimum distance classifier +LDA+ LATTICESVM
Recognition rate (%)	95.03	**98.01**

From the table, we can see that with our proposed two-stage method, the error rate decreases from 4.97% to 1.99%, which largely decreases by about **60%** ($\frac{4.97\%-1.99\%}{4.97\%}$). The experiment result illustrates the efficiency of the proposed algorithm.

V. CONCLUSION

For multi-class SVM methods, the OVA approach tends to have large training and testing time, while other competing approaches such as OVO and DAGSVM tend to store a large number of binary SVMs. The accuracy of the binary tree method is greatly influenced by the construction of the tree structure and by far no satisfying constructing approach is proposed. In this paper, a LATTICESVM algorithm is proposed, which successfully reduces the storage and computational burden and shows satisfying performance in recognition accuracy. A systematic analysis and the experimental results show that the LATTICESVM is an effective method for multi-class classification, especially for large-scale pattern recognition problems.

REFERENCES

[1]. Vapnik V, *The nature of statistical learning theory*. New York: Springer Press, 1995.

[2]. Vapnik, V, *Statistical Learning Theory*. John Wiley and Sons, Inc., New York, 1998.

[3]. Christopher J.C. Burges1, "A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, Springer Netherlands. Vol. 2, Number 2. June, 1998

[4]. Pavel Laskov, "Feasible direction decomposition algorithms for training support vector machines," Machine Learning, 46(1), 2002:315-349

[5]. Norikazu Takahashi, Tetsuo Nish, "Rigorous Proof of Termination of SMO Algorithm for Support Vector Machines". IEEE Transactions on Neural Networks, Vol. 16, No. 3, May 2005.

[6]. Debnath R, Takahide N, Takahashi H. “A decision based on one-against-one method for multi-class support vector machine,” in Pattern Anal Applic, 2004, 7:164-175.
[7]. U. Kreßel, “Pairwise classification and support vector machines,” in Advances in Kernel Methods—Support Vector Learning, B. Schölkopf, C.J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 255–268.
[8]. L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, et al. “Comparison of Classifier Methods: A Case Study in Handwriting Digit Recognition”, International Conference on Pattern Recognition. IEEE Computer Society Press, 1994, pp. 77-87.
[9]. J. C. Platt, N. Cristianini , J. Shawe-Taylor, “Large Margin DAGs for Multiclass Classification”, in Advances in Neural Information Processing Systems, volume 12, MIT Press, 2000, pp. 547-553.
[10]. Liu Zhigang, Shi Wenzhong,Qin Qianqing, Li Xiaowen, Xie Donghui, “Hierarchical support vector machines,” Geoscience and Remote Sensing Symposium, 2005. IGARSS, '05 Proceedings. 2005 IEEE International Volume 1, 25-29 July 2005
[11]. M. Azimi-Sadjadi ,S. Zekavat, “Cloud Classification Using Support Vector Machines”, IGARSS, 2000, pp. 669-671.
[12]. C. W. Hsu and C. J. Lin, “A Comparison of Methods for Multi-class Support Vector Machines,” IEEE Trans. Neural Networks, Vol. 13,no. 2, pp. 415-425, 2002.
[13]. J. Friedman. (1996) “Another Approach to Polychotomous Classification,” Dept. Statist., Stanford Univ., Stanford, CA. [Online]. Available:http://www-stat.stanford.edu/reports/friedman/poly.ps.Z
[14]. Heisele, B., Ho, P., Poggio, T., “Face recognition with support vector machines: global versus component-based approach” Computer Vision, 2001. ICCV 2001 Proceedings. Eighth IEEE International Conference on Volume 2, 7-14 July 2001 Page(s):688 - 694 vol.2
[15]. Liu Zhibin, Jin Lianwen. A Static Candidates Generation Technique and Its Application in Two-stage LDA Chinese Character Recognition. In: Proceedings of the 26th Chinese Control Conference, 2007
[16]. Platt J. “Fast training of support vector machines using sequential minimal optimization,” Advances in kernel methods: support vector learning [M]. Cambridge, U SA: The MIT Press, 1999: 185-208.
[17]. Liu Cheng-Lin, Nakashima, Kazuki, H, Sako, and H.Fujisawa. Handwritten digit recognition: investigation of normalization and feature extraction techniques. Pattern Recognition, Vol. 37, No.2, 2004.265-279
[18]. DUTA R. O., HART P.E.. *Pattern classification and Scene Analysis* . New York: John and Wiley, 1973.