2008 Eighth IEEE International Conference on Data Mining

Multi-Space-Mapped SVMs for Multi-Class Classification

Bo Liu, Longbing Cao
Faculty of Information Technology
University of Technology, Sydney, Australia
csbliu@gmail.com; lbcao@it.uts.edu.au

Philip S. Yu
University of Illinois, Chicago, USA
psyu@cs.uic.edu

Chengqi Zhang
Faculty of Information Technology
University of Technology, Sydney, Australia
chengqi@it.uts.edu.au

Abstract

In SVMs-based multiple classification, it is not always possible to find an appropriate kernel function to map all the classes from different distribution functions into a feature space where they are linearly separable from each other. This is even worse if the number of classes is very large. As a result, the classification accuracy is not as good as expected. In order to improve the performance of SVMs-based multi-classifiers, this paper proposes a method, named multi-space-mapped SVMs, to map the classes into different feature spaces and then classify them. The proposed method reduces the requirements for the kernel function. Substantial experiments have been conducted on One-against-All, One-against-One, FSVM, DDAG algorithms and our algorithm using six UCI data sets. The statistical results show that the proposed method has a higher probability of finding appropriate kernel functions than traditional methods and outperforms others.

1. Introduction

For the multiple algorithms with SVMs[1, 2], Vapnik [1] proposed One-against-All with a discrete decision function and then introduced the continuous decision function [3]. Inoue [4] presented fuzzy decision function, which is equivalent to a continuous decision. KreBel [5] proposed the One-against-One algorithm, but an unclassifiable region exists. Platt [6] put forward the Decision Directed Acyclic Graph (DDAG) decision function. Kijsirikul [7] proposed the Adaptive Directed Acyclic Graph (ADAG) algorithm, which is covered by DDAG. Tsujinishi [8] presented the fuzzy decision function.

In all algorithms based on SVMs, the purpose of the kernel function or mapping function is to embed the samples from the input space into a feature space, where the classes are expected to be found much more linearly separable [1]. If we observe the SVMs model in the feature space, it is actually a linear classifier [3]. Therefore, if the classes are nonlinearly separable in the feature space, the accuracy of the SVMs cannot be guaranteed. By this reasoning, an appropriate kernel function should be able to allow the classes to be linearly separable to the greatest extent. For the binary classification problem, a suitable kernel function can be found based on the theory of VC bound and Radius/Margin bound [3]. While for the multi-class classification problem, it is actually a great challenge to find such suitable functions. In fact, it is not always possible to find such an appropriate kernel function to make the classes linearly separable in the feature space, especially when the number of classes is very large. This is because all the classes come from different distribution functions, and there is not such theory bound as binary classification to indicate how to find a suitable kernel function. Thus, the accuracy of SVMs-based multiple classification is reduced, which could be drawn from [9].

In this paper, we propose a novel approach to the SVMs-based multi-class classification. By analyzing the three-class example with the VC dimension theory, we explain that the mapping of all of the classes into the same feature space is not suitable for multi-classification problems. Motivated by this point of view, this paper proposes a *multi-space-mapped* (MSM) SVMs. The proposed algorithm, different from all existing ones, presents a framework to classify all of the classes in different feature spaces based on a binary tree. Following MSM, one can first split the classes into two groups using the *progressive clustering algorithm*; the two groups are then mapped into a feature space as a binary classification problem, where they are mostly linearly

1550-4786/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDM.2008.13

separable; finally, we train the respective classifiers on each node of binary tree. Further, the Grid-Search and ten-fold cross validation techniques are adopted to tune the parameters of the SVMs.

Substantial experiments on six UCI datasets have been conducted to compare the performance of One-against-All, One-against-One, FSVM, and DDAG algorithms with the proposed method. The statistical results show that the MSM method outperforms other algorithms.

The rest of this paper is organized as follows. In Section 2, we review the typical multiple classification algorithms. The proposed MSM algorithm is detailed in Section 3. Experimental results is discussed in Section 4. Conclusions are drawn in Section 5.

2. Multiple Algorithms with SVMs

Let $S_{training} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$ be a training set, where $\mathbf{x}_i \in R^m$ and $y_i \in \{1, 2, \ldots, C\}$, We classify the typical multiple algorithms into the One-against-All and One-against-One schemes as follows.

2.1 One-against-All Scheme

One needs to determine C decision functions for $C-$ class problems. The optimal hyperplane for class i against the remaining classes is:

$$D_i(\mathbf{x}) = \mathbf{w}_i{}^T \phi(\mathbf{x}) + b_i = 0, \quad (1)$$

where $\mathbf{w}_i^T$ is a vector, $\phi(\mathbf{.})$ is a mapping function and b_i is a scalar. After this approach, C hyperplanes are obtained in the feature space. Three algorithms are derived as follows.

A *Discrete decision function discriminance*

For the input vector **x**, if a single i satisfies

$$D_i(\mathbf{x}) > 0, \quad (2)$$

then **x** is classified into class i. If multiple is satisfy the condition (2), or there is no i satisfying (2), then **x** is unclassifiable [1] as shown in Fig. 1.

B *Continuous decision function discriminance*

For the input vector **x**, it is classified into the class i if it satisfies

$$arg \max_{i=1,\ldots,C}(D_i(\mathbf{x})). \quad (3)$$

Since $D_i(\mathbf{x})$ is continuous, the decision function is continuous, and the unclassifiable regions are resolved [3] as shown in Fig. 2. Another algorithm based on the one-against-all scheme is the *fuzzy decision function*, which is equivalent to the continuous decision function method [4].

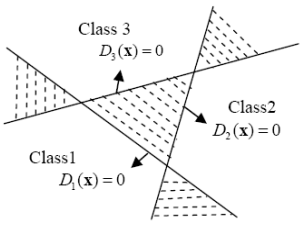

Figure 1. Discrete decision function discriminance.

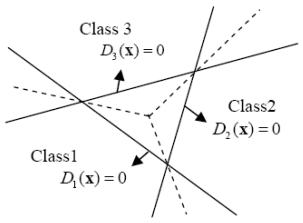

Figure 2. Continuous decision function discriminance.

2.2 One-against-One Scheme

One determines $C(C-1)/2$ decision functions for the $C-$ class problems [5]. The optimal hyperplane for class i against class j is:

$$D_{ij}(\mathbf{x}) = \mathbf{w}_{ij}^T \phi(\mathbf{x}) + b_{ij} = 0, \ i < j, 1 < j \leq C, 1 \leq i < C,$$

where $\mathbf{w}_{ij}^T$ is a vector, $\phi(.)$ is a mapping function, and b_{ij} is a scalar. And then (4) is defined.

$$D_{ij}(\mathbf{x}) = -D_{ji}(\mathbf{x}). \quad (4)$$

Three methods are derived based on this scheme.

A *One-against-One discriminance*

For the input vector **x**, if the following condition is satisfied

$$D_i(\mathbf{x}) = \sum_{j \neq i, j=1}^{C} sgn(D_{ij}(\mathbf{x})), \quad (5)$$

then **x** can be classified into the class

$$arg \max_{i=1,\ldots,C}(D_i(\mathbf{x})). \quad (6)$$

If i is not unique in (6), then **x** is unclassifiable as shown in Fig. 3.

B *Decision directed acyclic graph (DDAG) discriminance*

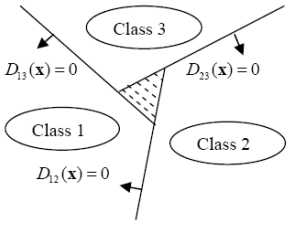

Figure 3. One-against-One discriminance.

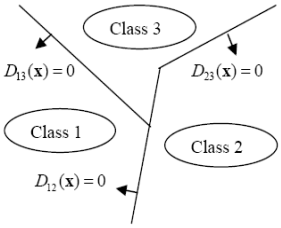

Figure 4. The boundary of DDAG.

DDAG [6] was introduced to handle the unclassifiable region. At the top-level classification, one can choose any pair of classes. Except for the leaf node, if $D_{ij}(\mathbf{x}) > 0$, then one can regard $\mathbf{x}$ to not belong to class j. If $D_{12}(\mathbf{x}) > 0$, it means $\mathbf{x}$ does not belong to class 2, thus it belongs to either class 1 or 3. Therefore the next classification pair is classes 1 and 3. Using this method, the unclassifiable region is resolved, but it depends on the tree information. The unclassifiable region is assigned to the classes associated with the leaf node as shown in Fig. 4.

C *Fuzzy decision function discriminance*

For the input vector $\mathbf{x}$, $m_{ij}(\mathbf{x})(i, j = 1, 2, ..., C)$ is defined as follows:

$$m_{ij}(\mathbf{x}) = \begin{cases} 1 & 1 \leq D_{ij}(\mathbf{x}) \\ D_{ij}(\mathbf{x}) & otherwise \end{cases}. \tag{7}$$

The membership function $m_i(\mathbf{x})$ is given by:

$$m_i(\mathbf{x}) = \min_{j=1,...,C}(m_{ij}(\mathbf{x})). \tag{8}$$

Using (8), sample $\mathbf{x}$ is classified into the class

$$\underset{i=1,...,C}{argmax}(m_i(\mathbf{x})). \tag{9}$$

Based on (9), the unclassifiable region could be handled as shown in Fig. 5.

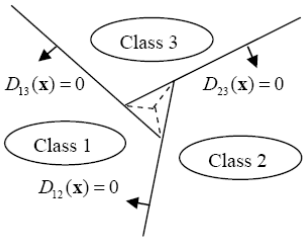

Figure 5. The fuzzy decision function discriminance.

3. Multi-Space-Mapped SVMs for multi-class classification

3.1. A framework for three-class problems

Assume Classes 1, 2 and 3 are nonlinearly separable in the input space as shown in Fig. 6 (a). To resolve the problem using traditional SVMs-based algorithms, the three classes are mapped into a feature space, where they are linearly separable. The VC bound can be denoted as R^2/Δ^2 [3], where Δ is the $\Delta-$ margin of hyper-plane, R is the radius of the minimum ball including all data points in the feature space. We should minimize R^2/Δ^2 to obtain good performance by minimizing R and maximizing Δ. For R, if the kernel function is confirmed, the distribution of classes is formed and R is known. For Δ , its maximal value is equal to $2/\|\mathbf{w}\|$. Thus, R^2/Δ^2 mainly depends on kernel functions, and the better kernel function should minimize R^2/Δ^2. From the VC dimension of an SVM, we can obtain the following analytic results.

(1) For Classes 1 and 2, suppose we find the kernel function $K_{12}(.,.)$ which minimizes R_{12}^2/Δ_{12}^2. In the corresponding feature space Classes 1 and 2 are most linearly separable.

(2) For Classes 1 and 3, suppose we obtain the kernel function $K_{13}(.,.)$ which minimizes R_{13}^2/Δ_{13}^2. In the feature space, Classes 1 and 3 are most linearly separable.

(3) As for Classes 2 and 3, assume we find the kernel function $K_{23}(.,.)$ which can minimize R_{23}^2/Δ_{23}^2, and then in the feature space Classes 2 and 3 are most linearly separable.

Generally speaking, the three kernel functions $K_{12}(.,.)$, $K_{13}(.,.)$ and $K_{12}(.,.)$ are not equivalent, since the three classes always come from different distributions. Thus, it would be better to map the three classes into different feature spaces and then classify them. The idea of the proposed three-class problem is presented as follows:

(1) Choose the two classes most similar to each other from the three classes and consider them as set 1; consider

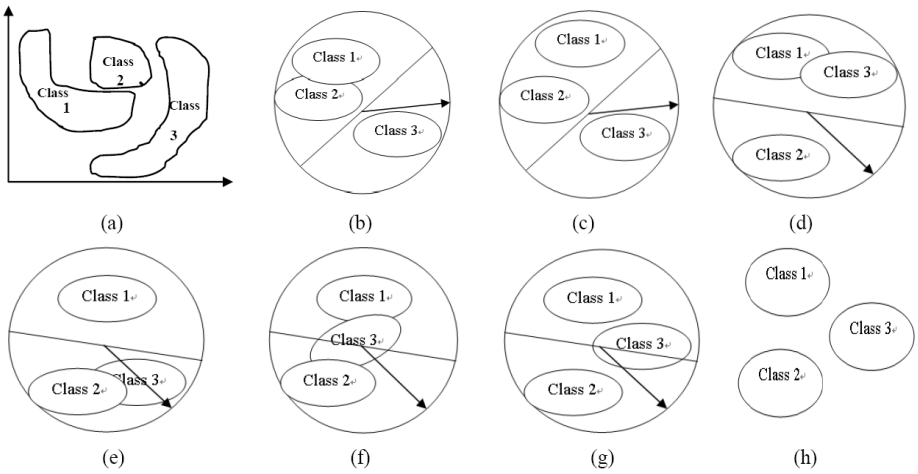

Figure 6. The three-class example.

the other as set 2. Assume set 1 contains classes 1 and 2, and class 3 belongs to set 2. (2) Treat sets 1 and 2 as a binary classification problem, select a proper kernel function which makes sets 1 and 2 linearly separable in the corresponding feature space. Then, construct the hyperplane using SVMs to separate sets 1 and 2. At this stage, we do not care whether classes 1 and 2 are linearly separable or not. In fact, they may cross with each other (see Fig. 6 (b)) or be linearly separable (Fig. 6 (c)) in this feature space. (3) Consider the two classes in set 1 as a binary classification, and map them to another feature space using another kernel function, which makes them linearly separable, then construct a hyperplane with SVMs to classify them. In this process, class 3 is not considered, while it could cross with class 1 (Fig. 6 (d)), class 2 (Fig. 6 (e)), classes 1 and 2 together (Fig. 6 (f)), or linearly separable (Fig. 6 (g)).

The proposed idea can be realized through the use of a binary tree. The characteristics of the proposed framework are: (1) The proposed idea reduces the requirements for the kernel function, and has a higher probability of finding the appropriate kernel functions than traditional algorithms; (2) If an appropriate kernel function $K_1(.,.)$ is identified so that the three classes are linearly separable in the corresponding feature space (as shown in Fig. 6 (h)), the traditional algorithms can achieve good performance. In this case, kernel functions $K_1(.,.)$ can also be used in the proposed method and result in good performance as well.

3.2. The multi-Space-Mapped SVMs

In the multiple classification which classes come from different distribution functions, if there is not a kernel function or too complicated to select a kernel function in making all classes linearly separable in a feature space, the performance of the algorithms would be low. To handle such issue, we propose a *multi-space-mapped* SVMs algorithm.

3.2.1 Preparation Methods

Given three sets S, S_1 and S_2, suppose we have m classes to be partitioned, then store them into set S as follows.

$$S = \{Class_1, Class_2, \ldots, Class_m\}, \tag{10}$$

let S^i represent the i^{th} element in S, then $S^i = Class_i$. We further separate the classes in S into two subsets and assign them into sets S_1 and S_2 respectively. Particular attention must be paid to the fact that that all samples belonging to the same class are assigned together, and the whole class is treated as an entity. The K-means method is then used (K=2). We continue to define the following notions.

The mean m_i^* and the within-set scatter SC_i^* of sets 1 and 2 are defined as follows.

$$m_i^* = \frac{1}{l_i^*} \sum_{j=1}^{l_i^*} \mathbf{x}_{ij}^*, \quad i = 1, 2 \tag{11}$$

$$SC_i^* = \frac{1}{l_i^*} \sum_{j=1}^{l_i^*} \sum_{k=1}^{l_i^*} \|\mathbf{x}_{ij}^* - \mathbf{x}_{ik}^*\|^2 \quad i = 1, 2, \tag{12}$$

where l_i^* is sample size of set i , and $\mathbf{x}_{ij} \in Set\ i$. Similarly we get the following for all classes:

$$m_i = \frac{1}{l_i} \sum_{j=1}^{l_i} \mathbf{x}_{ij}, \quad i = 1, 2, \ldots, C \tag{13}$$

$$SC_i = \frac{1}{l_i} \sum_{j=1}^{l_i} \sum_{k=1}^{l_i} \|\mathbf{x}_{ij} - \mathbf{x}_{ik}\|^2 \quad i = 1, 2 \ldots, C, \tag{14}$$

where m_i and SC_i represent the mean and the within-class scatter of Class i respectively, l_i is the sample size of class i, and $\mathbf{x}_{ij} \in Class\ i$.

The target function J_{ij} is defined as:

$$J_{ij} = \frac{\|m_i^* - m_j\|^2}{SC_i^* + SC_j} \quad i = 1, 2;\ j = 1, 2, \ldots, C, \qquad (15)$$

where J_{ij} implies the similarity between set i and class j. The smaller J_{ij}, the more similar set i and class j. Let *Count(S)* refer to the number of classes in set S, the *Progressive K-means Algorithm* is proposed to split the classes in S into S_1 and S_2 as follows.

Algorithm *Progressive K-means Algorithm*

1. Input: $S \longleftarrow$ {the classes to be split};
2. Initialize: $S_1 \longleftarrow \emptyset$, $S_2 \longleftarrow \emptyset$;
3. Define arrays m^T, S^T; Num=*Count(S)*;
4. **for** (i=1; i++; i $\leq$ C){
5. $m_i^T \longleftarrow m_i$ {According to (13)}
6. $S_i^T \longleftarrow SC_i$ {According to (14)}
7. }
8. Choose two classes randomly from S, and assign them into S_1 and S_2 respectively;
9. Compute m_i^*, SC_i^* of S_1 and S_2 based on (11) (12);
10. **for** $(j = 1; j++; j \leq Num)$ {
11. **if** class j does not belong to S_1 or S_2;{
12. Compute J_{1j}, and J_{2j} based on (15);
13. **if** $J_{1j} < J_{2j}$ {
14. $S_1 = S_1 \bigcup$ Class j;
15. }
16. **else** {
17. $S_2 = S_2 \bigcup$ Class j;
18. }
19. }
20. }
21. Return (S_1, S_2)

After using the above algorithm, all the classes in S_1 are combined as one class with label $+1$ and the classes in the set S_2 are regarded as one class with label -1. One then constructs a hyperplane with SVMs to separate S_1 and S_2. In order to guarantee the better performance of the classifier, a finer kernel function is required to map S_1 and S_2 to a feature space, where they are linearly separable.

In SVMs, the Grid Search [9] method is always used to determine better kernel parameters for one type of kernel function as follows.

Assume the parameter set is $\pi = \{\pi_1, \pi_2, \ldots, \pi_{mun}\}$, mun is the number of parameters. In π, one parameter is from SVMs, and the remaining $m-1$ parameters come from the kernel function. Suppose π_i have n_i choices, then there are in total $n_{Total} = n_1 * n_2 * \ldots * n_{mun}$ choices.

Grid-Search and the ten-fold cross validation techniques are used to tune the parameters (details in Section 4). With the tuned parameters, a hyperplane can be constructed to separate the two sets with a higher accuracy.

3.2.2 The Multi-Space-Mapped SVMs

Define an array *TStore*, whose element *TStore*$_i$ stores the classes to be partitioned into S_1 or S_2. The *multi-space-mapped* SVMs based on a binary tree is described as follows.

Algorithm *Multi-Space-Mapped SVMs Based on SVMs*

1. Input: A training set $S_{training}$, the given type of kernel function;
2. **for** $(i = 1; i++; i \leq 2 \cdot C - 1)$ {
3. *TStore*$_i \longleftarrow \emptyset$;
4. }
5. *TStore*$_1 \longleftarrow$ {all classes in $S_{training}$};
6. j=1;
7. **for** $(i = 1; i++; 2 \cdot C - 1)$ {
8. $S \longleftarrow TStore_i$;
9. **if** $Count(S) > 1$ {
10. Use progressive K-means algorithm to split S into S_1 or S_2;
11. Tune the optimal parameters, construct SVMs to separate S_1 and S_2; {Note: one node of the binary tree is obtained}
12. $j = j + 1, TStore_j \longleftarrow S_1$;
13. $j = j + 1, TStore_j \longleftarrow S_2$;
14. }
15. **if** $Count(S) = 1$ {
16. The class in S is a leaf of the binary tree.
17. }
18. }

In executing this algorithm, a binary tree is generated, which contains $C - 1$ nodes and C leaves. For this reason, the range of the integer variable i is set [1, $2 \cdot C - 1$] in the above algorithm.

The proposed method is different from other algorithms based on decision trees [10, 11]. In these methods, similar to One-against-All and One-against-One scheme, the dataset with all the classes are mapped into the same feature space, where a decision tree is constructed with SVMs.

4. Experiment

PLS, Auto-Mpg (AM), Balance Scale (BS), Vehicle (VE), Vowel (VO), and Page-blocks (PB) dataset in UCI machine learning repository [12] were used to compare the performance of One-against-All with continuous decision function (called One-against-All in brief), One-against-One, FSVM, DDAG and our proposed method. The RBF kernel function (16) was utilized in the experiment.

Table 1. The accuracies of the datasets.

datasets	O-A-A	O-A-O	DDAG	FSVM	MSM
PLS	0.9114	0.9023	0.9084	0.9054	0.9252
AM	0.8723	0.8667	0.8718	0.8744	0.9029
BS	0.9841	0.9824	0.9839	0.9855	0.9968
VE	0.8558	0.8508	0.8556	0.8544	0.8817
VO	0.9869	0.9768	0.9909	0.9909	0.9960
PB	0.9753	0.9744	0.9753	0.9750	0.9804

$$K(\mathbf{x}, \mathbf{x}_i) = exp(-\|\mathbf{x} - \mathbf{x}_i)\|_2^2/\sigma^2) \qquad (16)$$

As a preprocessing step for each dataset, all records containing unknown values were removed, and each component of the $\mathbf{x}_i$'s was normalized to $[-1, 1]$. Ten-fold validation was used to split each dataset into ten pairs of training and testing sets. Furthermore, ten-fold validation was utilized again to tune the parameters for each pair of training and testing sets. In all algorithms, γ and σ come from SVMs and RBF function respectively. Let γ and σ have ten choices: $\sigma = [2^{-4}, 2^{-3}, 2^{-2}, \ldots, 2^5]$, $\gamma = [2^1, 2^2, 2^3, \ldots, 2^{10}]$. From Table 1, we find that the proposed method outperforms the other algorithms at all times.

In the traditional algorithms, all classes are mapped into the same feature space using a kernel function or $\phi(.)$, where they need to be linearly separable. In our method, we split the dataset into two subsets on each node of the binary tree, and tune the optimal parameters for each node. Even if the two sets are linearly separable in the feature space, the classes in the same set do not need to be linearly separable. Therefore, the requirement for the kernel function is greatly reduced by the proposed method compared with the other algorithms.

5. Conclusions

The performance of SVMs-based multiple classification is subject to the selection of kernel functions. To release the requirement on kernel functions, this paper has proposed a *multi-space-mapped* SVMs that demonstrates improved performance in multiple classification problems. In this method, all classes are mapped into different feature spaces and then classified. Extensive experiments of One-against-All, One-against-One, FSVM, DDAG and MSM method have been conducted on six UCI datasets. The results show that MSM provides better performance as well as increased flexible requirements on kernel functions than the others.

In fact, the MSM approach presents a generic framework for multi-classification by first partitioning all classes into two groups that are further mapped to a feature space, and then training the respective classifiers. Therefore, the proposed technique is very useful for tackling difficulties in traditional multi-class classification problems using either SVMs or other classifiers.

The essence of our method is to classify the classes in different feature spaces. We will apply this approach to the kernel method and other machine learning tools.

References

[1] V. N. Vapnik, "The nature of statistical learning theory," Springer-Verlag, London, UK, 1995.

[2] B. Liu, Y. Dai, X. Li, W. S. Lee, P. S. Yu: " Building text classifiers using positive and unlabeled examples," ICDM 2003, pp. 179-188.

[3] V. N. Vapnik, "Statistical learning theory," John Wiley, Sons, 1998.

[4] T. Inoue, S. Abe, "Fuzzy support vector machines for pattern classification," *IJCNN 2001*, pp. 1449-1454.

[5] U. H. G. KreBel, "Pairwise classification and support vector machines," *Advances in Kernel Methods: Support Vector Learning*, pp. 255-268, 1999, the MIT Press.

[6] J. C. Platt, N. Cristianini, J. Shawe.-Taylor, "Large margin DAGs for multiclass classification," *Advances in Neural Information Processing Systems* vol. 12, pp. 547-553, 2000, the MIT Press.

[7] B. Kijsirikul, N. Ussivakul, "Multiclass support vector machines using adaptive directed acyclic graph," *IJCNN*, pp. 980-985, 2002.

[8] D. Tsujinishi, S. Abe, "Fuzzy least squares support vector machines for multiclass problems," *Neural Network*, vol. 16, pp. 785-792, 2003.

[9] C. W. Hsu, C. J. A. Lin, "Comparison of methods for multiclass support vector machines," *IEEE Transaction on Neural Networks*, vol. 13, pp. 415-425, 2002.

[10] N. Cesa-Bianchi, C. Gentile, L. Zaniboni ,"Incremental algorithms for hierarchical classification," *JMLR*, vol. 7, pp. 31-54, 2006.

[11] G. Zhang, and W. Jin,"Automatic construction algorithm for multi-class support vector machines with binary tree architecture," *International Journal of Computer Science and Network Security*, vol. 6, no. 2A, 2006.

[12] P. M. Murphy, UCI database, http://www.ics.uci.edu/ mlearn /MLRepos itory.html, 2004.