GP-SVM: Tree Structured Multiclass SVM with Greedy Partitioning

Sandeep Kumar Sahu*, Arun K. Pujari†, Venkateswara Rao Kagita‡, Vikas Kumar§, Vineet Padmanabhan¶
Email: {*uusandeepsahu, §vikas}@uohyd.ac.in, ‡585venkat@gmail.com, {†akpcs, ¶vineetcs}@uohyd.ernet.in
School of Computer & Information Sciences, University of Hyderabad, India-500046

***Abstract*—In this paper, we propose a hierarchical SVM framework for multiclass classification problems. Use of multiple SVMs in a hierarchical structure has been a popular approach to handle multiclass classification by Support Vector Machines which are otherwise known to two-class classifiers. Among commonly-used hierarchical structures, binary tree structured SVM has computational advantages over other techniques. In order to devise an effective tree structured hierarchy of multiple SVMs, it is important to devise a process of recursive subdivision of classes, known as binarization process. We propose here a greedy heuristic as binarization strategy with partition function as the separability measure. To the best of our knowledge, no attempt has been made in this direction and the proposed algorithm takes advantage of partition function, binary structure and class membership information. We show empirically that our method provides higher accuracy with less computational overhead compared to most of the major multiclass SVM classifiers. Our method is also useful in taxonomy learning for multiclass problems.**

I. Introduction

Extension of two-class classifiers, SVMs in particular, to handle multiclass classification problems is well studied, yet considered an open issue [1], [2]. There are two major approaches for extension of binary classifiers to multiclass case. The first approach explicitly reformulates the classifier, resulting in a unified (single machine) multiclass optimization problem (*embedded techniques*) [3], [4], [5], [6]. The second approach is to decompose a multiclass problem into multiple, independently trained, binary classification problems and to combine them appropriately so as to form a multiclass classifier (*combinational techniques*). Combinational techniques have been very popular and successful as they all exhibit some relative advantages over embedded techniques. Some major combinational techniques are OAA [2], OAO [7], [8], DAGSVM [1] and BT-SVM [9], [10]. All these techniques arrange SVMs in a hierarchical fashion and the efficiency is measured in terms of training complexity and testing complexity besides the accuracy of multi-class classification. It is observed in comparative studies of combinational techniques that by and large similar accuracy can be obtained by these techniques if the parameters are tuned suitably and hence preference of one method over another is essentially on computational aspects such as on the number of binary SVMs generated, on the number of SVMs required to be evoked during classification or, on the size of training set for each SVM. Tree structured SVMs is considered to be better than other techniques in terms of these parameters. However, one of the major research issues in tree-structured SVM is to devise a recursive subdivision scheme of classes to obtain the structure. When the tree is binary, this problem is known as binarization process which is an active area of research.

In this paper, we propose a new and efficient tree-structured hierarchical SVM for multi-class classification, termed as *GP-SVM*. We use *partition function* as a separating measure for the classes and use a greedy heuristic to construct a set of binary partitions. An important aspect of our technique is that we make efficient use of individual training patterns as well as their class information. Through rigorous experimentation, we demonstrate several advantages of our method. In [11], the concept of partition function is used to propose an OAA-based tree structured classification. For several benchmark datasets, the method outperforms all major combinational methods such as OAA, OAO and DAGSVM. We show that our method yields better accuracy than that of OAA-based method [11], particularly when the number of classes is large. Due to binary-tree structure, our method is computationally better than the OAA-based method. Furthermore, for datasets with large number of classes, our method gives significantly better accuracy than the earlier method. We also show that hierarchical taxonomy can be learnt in the process of binary-tree construction. Similar classes are grouped together in the hierarchy and a delayed classification takes place for this group in a separate branch of the tree. The user can have an option of grouping these into one class and can obtain higher classification accuracy at the expense of combining similar classes together.

Rest of the paper is organized as follows. Section II describes different combinational methods for multiclass classification. In Section III, earlier work related to binary tree structured SVM is reviewed. In Section IV, we propose greedy based partition approach for efficient and accurate classification. Experimental analysis of the proposed algorithm is reported in Section V and we conclude with Section VI.

II. Combinational Methods for Multiclass Classification

In this section, we give a brief review of the major combinational techniques for the sake of completeness. There are several reviews and survey papers on multi-class SVMs. A critical review of all multiclass methods can be found in [12].

A. OAA (One-Against-All)

One-against-all SVM (OAA-SVM) is one of the earliest and simplest multiple-machine approaches. For the K-class problems ($K > 2$), K two-class SVM classifiers are constructed [13], [2]. The i^{th} SVM is trained with the samples in the i^{th} class as positive examples and all the rest as negative

978-1-5090-0487-4/15 $31.00 © 2015 IEEE
DOI 10.1109/ICIT.2015.24

examples. In the recognition phase, a test example is presented to all K SVMs and is labeled according to the maximum output among the K classifiers. The disadvantage of this method is that each of the K classifiers is trained using all available samples. Recently, a reduced R-OAA-SVM [14] is proposed based on the subset sample selection to decrease the training and testing time of OAA-SVM.

B. OAO (One-Against-One)

OAO techniques train $\frac{K(K-1)}{2}$ binary classifiers, one for each pair of classes, for K class problems ([7], [15]). A test sample is tested with all SVMs and majority voting is typically employed for testing. These $\frac{K(K-1)}{2}$ binary classifiers differ in size of the training set. Assuming, for simplicity, that each classifier has an equal amount of training samples $\frac{2N}{K}$, the training complexity of OAO is far smaller than that of OAA. The testing stage requires examination of $\frac{K(K-1)}{2}$ binary classifiers.

C. DAG-SVM

The DAGs (Directed Acyclic Graphs) technique proposed by Platt et al. [1] is an alternative way of OAO technique. DAGSVM is a directed graph with no cycles and, at most two edges pointing to the same node. More precisely, the nodes of a DAG are arranged in a triangle with a single root node at the top, two nodes in the second layer and so on down to the final layer. Like OAO technique, DAG-SVM trains all $\frac{K(K-1)}{2}$ binary classifiers. If a classifier decides one class (say, positive class) then the other class (negative class, in this case) is rejected. Through a series of successive disqualifications of candidate decision classes, the final decision is reached. DAGs require $K-1$ examinations of binary classifiers to reach the final decision, in comparison to the $\frac{K(K-1)}{2}$ examinations required for OAO testing with majority voting. One disadvantage of DAG-SVM is that its result depend on the predefined order in which the binary classifiers are picked up during testing. Another drawback is that the efficiency depends on the order of the classes appearing in DAG.

D. BT-SVM: Binary-Tree Structured SVMs

It is natural to organize the SVMs as a binary tree. It has advantages of both the efficient computation of the tree architecture and the high classification accuracy of SVMs. With this architecture, $K-1$ SVMs are necessary during training of a K-class problem and only $log_2 K$ SVMs in average are required to classify an unknown sample. This leads to a dramatic improvement in recognition speed for problems with large K. In this formulation, both the training and testing phases are carried out in a top-down manner and the classes are divided into two disjoint subsets at each node and a SVM is trained. The training starts with the whole dataset and at any stage the current dataset is partitioned recursively into two non-overlapping subsets. These two subsets are used as positive and negative samples to train a SVM classifier associated with the node. For classification, any unknown test pattern traverses the tree top-down and at every node, based on the response of the corresponding SVM, the test pattern is passed to one of the branches (positive or negative) of the tree. Table I gives comparative analysis of complexities of different combinational methods [16].

TABLE I: Comparison of combinational methods for K-class with N number of training samples.

Method	#classifiers	#queries	#samples/classifier
OAA	K	K	N
OAO	$\frac{K(K-1)}{2}$	$\frac{K(K-1)}{2}$	$\frac{2N}{K}$
DAG-SVM	$\frac{K(K-1)}{2}$	$K-1$	$\frac{2N}{K}$
BT-SVM	$K-1$	$log_2(K)$ $\frac{K-1}{2}$	$\frac{Nlog_2(K)}{K-1}$ balanced $< \frac{N(K-2)}{K-1}$ unbalanced

One major issue of tree-structured classifiers is the problem of binarization. *Binarization* is the strategy to divide the classes into two non-overlapping subsets. There are several multiclass classifiers adopting the binary tree structure and they differ among themselves in the binarization method they employ. In [9], it is proposed to group confusion classes in agglomerative framework. Divide-by-2 method of Vural and Dy [17] and Half-Against-Half method of Lei and Govindaraju [18] are some of the first attempts in using hierarchical divisive clustering. Cheong et al. [19] use a Kernel based SOM(KSOM) for binary partition and Chen et al. [20] use max-cut technique with KL-divergence between feature distribution from different classes. Vural et al. [17], Liu et al. [21] and Yuan et al. [22] employ K-Means clustering to group the classes into two subsets at every node. Lorena et al. [23] use the concept of Minimum Spanning Trees for binarization. In [10], authors propose to use centroids as representative of classes and two classes are selected and assign all the other examples to the nearest centroid. Madzarov et al. [24] use farthest distance between class-centroids for divisive clustering. Bengio et al. [25] use confusion matrix as an affinity matrix between classes and use spectral clustering. Liu et al. [26] use one class SVM to estimate pairwise distance between classes. In [27], class hierarchies are constructed by postponing some decisions using uncertainty and resulting in higher recognition accuracy. In [28], each partition is solved by an optimized SVM and the one with the best performance is chosen at every node of the tree.

III. OAA-PARTITION FOR BINARIZATION

Based on the foregoing discussion, we understand that binary-tree structured multiclass classifiers are computationally efficient and the crucial design issue is binarization or the recursive partitions of set of classes. A partition is optimal when the gap between two components is the largest. Recently, in [11], partition functions are used as measures of separation between sets of training patterns and the set is partitioned by considering the highest value of the function. There are four alternative partition functions proposed by the authors and for the present study, we use only one of the functions.

Let $\mathbf{C} = \{C_1, C_2, \ldots, C_K\}$ be the set of classes, $X = \{x_1, x_2, \ldots, x_N\}$ be the samples ($x_i \in R^d$). Let X_i denote the set of samples in class C_i. The idea is to construct a tree structure of SVMs in which each internal node trains one SVM that classifies the sample into one of two groups.

Let $I \subset X$ be a set of training samples. A partition of I is defined as I_1 and I_2 with $I = I_1 \cup I_2$ and $I_1 \cap I_2 = \emptyset$.

Partition Function for a partition of I is defined as follows:

$$PF(I_1, I_2) = \frac{dist(c_1, c_2)}{S_1 + S_2}$$

where c_i is the center of samples in I_i, $dist(c_1, c_2)$ is Euclidean distance between c_1 and c_2, and S_i, the within-set scatter of samples in I_i, is defined as follows.

$$S_i = \frac{1}{l_i} \sum_{j=1}^{l_i} \sum_{k=1}^{l_i} ||x_j^i - x_k^i||^2, i = 1, 2,$$

where $l_i = |I_i|$ and $x_j^i \in I_i$. Higher value of $PF(I_1, I_2)$ indicates better separation between I_1 and I_2.

A partition of I as I_1 and I_2 is said to be an *OAA-partition* if I_1 is a single class and $I_2 = I \setminus I_1$. The algorithm in [11] begins with $I = X (= \cup_{i=1}^{K} X_i)$ at the root level and recursively partitions X, every time choosing the best OAA-partition. The class corresponding to the best OAA-partition at one stage is not considered in the subsequent stages. At any stage s, for a given I^s, it selects the class C^s corresponding to the best OAA-partition as follows.

$$C^s = \underset{C_i \in \mathbf{C}}{argmax}\ PF(X_i, I^s \setminus X_i)$$

Determining the best OAA-partition requires computational time linearly proportional to the number of classes in I^s. A two-class SVM is trained at this stage with data points in X^s as negative exemplars and $I^s \setminus X^s$ as positive exemplars. The process is repeated for stage $s+1$ with I^{s+1} as $I^s \setminus X^s$ till I contains all elements of only one class. At every stage a two-class SVM is trained with the single class versus the remaining set of classes at that stage. For a K-class problem, $K-1$ SVMs are trained. The size of training set decreases as the recursive procedure progresses. For classification, an unknown test pattern traverses top-down till it is classified as a negative pattern by one of the SVMs. Thus, testing requires at most $K-1$ SVMs to be revoked.

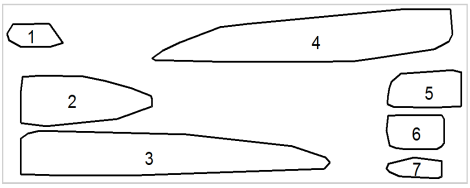
For the present discussion, we assume that SVMs used at every stage are properly designed and tuned with respect to the kernels and hyperparameters.

IV. GP-SVM - Our Method

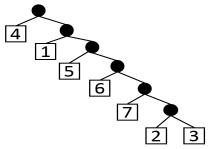
We observe in Table I that balanced tree structure is the most efficient. OAA-partition is very imbalanced as it expands only one node at every stage. In this section, we first examine the pros and cons of extending the OAA-algorithm to balanced tree structure. A partition of I as I_1 and I_2 is said to be a *true binary partition* if I_1 is a subset of I and $I_2 = I \setminus I_1$. The best possible scenario would be to have half of the classes of I in I_1 and other half in I_2. The above OAA-algorithm can be extended for *true-binary-partition* as follows. We start with root node with $I = X = \mathbf{C}$. At any stage s, for a given I^s, it selects $\mathbf{C}^s$, a subset of classes, corresponding to the best binary partition as follows.

$$\mathbf{C}^s = \underset{\mathbf{C}' \subseteq \mathbf{C}}{argmax}\ PF(X', I^s \setminus X')$$

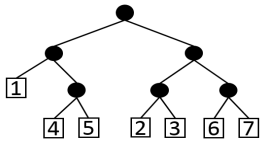
where X' is the set of training samples in subset of classes $\mathbf{C}'$.



(a) Boundaries of seven classes



(b) OAA-partition



(c) Exhaustive enumeration

Fig. 1: (a). Outlines of 7 classes in 2-dimensional space; (b). OAA-partition tree and (c). Exhaustive enumeration tree. Leaf nodes indicate class labels.

A two-class SVM is trained at this stage with data points in $\mathbf{C}^s$ as negative exemplars and remaining as positive exemplars. We thus generate two subproblems, unlike the earlier case, one corresponding to $\mathbf{C}^s$ and the other corresponding to $I^s \setminus \mathbf{C}^s$. The process is repeated recursively for both the subproblems. Like the earlier case, we have $K-1$ SVMs for a K-class problem but these are structured as a binary tree. Determining the best binary partition amounts to searching all possible subsets of classes at every stage and requires exponential time in number of classes in I^s. Nevertheless, a *true-binary-partition* is expected to yield better classifier than a *OAA-partition*. To see this, let us consider the following simple motivating example of 7 classes in 2-dimension (Figure 1a). There are 1207 points but the outlines of each class are depicted for the sake of clarity. OAA partition [11] yields tree as shown in Figure 1b, whereas with true binarization we get a different binary tree structure (Figure 1c). By training SVMs with Linear kernel and using random test patterns, we observe that the accuracy of classification for true binary tree is 100% whereas that of OAA-partition is 94%. The height of the tree in case of exhaustive enumeration is smaller as compared to OAA-partition.

Determining the best binary partition by exhaustive enumeration is not practical. The present work has the following motivation. Can there be an efficient method to compute true binary-tree structure using partition function as the separating measure? And if so, will it result in a more accurate classifier? Can it handle massively large number of classes? It may be noted, at this stage, that majority of the research papers published on this topic report experimental results on benchmark data having few number of classes. In this paper, we are concerned with research problems addressed above and provide affirmative solutions to these.

GP-SVM - We propose here a greedy heuristic to determine the best binary partition which improves the classification accuracy and at the same time, less demanding on the computational overhead. We show that our algorithm yields more accurate classifier for problems with large number of classes.

A *true binary partition* of I as I_1 and I_2 is said to be a *greedy partition* if I_1 is constructed sequentially in a greedy manner. In other words, we use a sequential constructive procedure that builds up the set I_1 class by class each time selecting a class of elements using a greedy strategy. We elaborate this step below. Starting with the empty set, at step t of greedy method, we add class C_t to I_1^t by selecting

$$C_t = \underset{C_i \in \mathbf{C}}{argmax}\ PF(I_1^t \cup X_i, I_2^t \setminus X_i)$$

where I_1^t is partially constructed I_1 at step t and I_2^t is the complement. We update I_1^t by adding C_t if

$$PF(I_1^t \cup X_t, I_2^t \setminus X_t) > PF(I_1^t, I_2^t)$$

otherwise, the greedy process terminates.

In a similar manner as *OAA-partition* and *true binary partition*, GP-SVM starts with $\mathbf{C}$ at the root level. It makes use of greedy partition at every stage and a two-class SVM is learnt. Like binary partition, two subproblems are generated for the next stage of recursion. This process requires $K-1$ SVMs for K-class problem.

We demonstrate the working of the process for the toy example given in Figure 1. Initially we start with X and at every stage s of true-binary parttion, we have I^s which is obtained by a greedy process. Thus we start with $\mathbf{C} = \{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_7}\}$. At first stage, $s = 1$, we employ greedy process as follows. $PF(X_i, I \setminus X_i)$ is calculated for each class C_i and the highest PF value corresponds to C_4. For greedy stage t, we have $I_1^t = X_4$ and $I_2^t = \mathbf{C} \setminus \mathbf{I_1^t}$. At the next step of greedy process, we find $C_t = \underset{C_i \in \mathbf{C}}{argmax}\ PF(I_1^t \cup X_i, I_2^t \setminus X_i)$ by considering $\{C_1, C_4\}, \{C_2, C_4\}, \{C_3, C_4\}, \{C_4, C_5\}, \{C_4, C_6\}$ and $\{C_4, C_7\}$. PF value corresponding to $\{C_1, C_4\}$ is the highest and it satisfies $PF(I_1^t \cup X_t, I_2^t \setminus X_t) > PF(I_1^t, I_2^t)$ with $I_1^t = X_4$ and $X_t = X_1$. The greedy process yields $\{C_1, C_4, C_5\}$ at the next step. Proceeding to the next step, we consider supersets of $\{C_1, C_4, C_5\}$ but as the terminating condition is satisfied, the greedy process returns the best partition as $\mathbf{C^s} = \{\mathbf{C_1}, \mathbf{C_4}, \mathbf{C_5}\}$. We repeat the same procedure for the next stage by considering two subproblems and the partition as shown in Figure (1c) is obtained. Though greedy can not guarantee the optimal partition all the time, our empirical analysis shows that greedy is as good as brute-force (or, exhaustive enumeration) in most of the cases.

V. Experimental Analysis

We evaluate the proposed algorithm on 14 benchmark datasets commonly used in the studies of multiclass classification. The detailed description of these datasets are given in Table II. First 9 datasets of Table II and Letter dataset are taken from UCI-ML repository (https://archive.ics.uci.edu/ml/datasets.html). Yale and ORL are two real world face recognition datasets taken from http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html. USPS dataset is from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html and Collins dataset from http://lib.stat.cmu.edu/datasets/. Rate of classification accuracy is used as the performance evaluation metric and testing time as efficiency metric. We compared our proposed approach with most recent work on partition function proposed in [11], referred as *OAA-partition* method in the subsequent discussion. It is shown in [11] that *OAA-partition* is better than all major combinational techniques. We use LIBSVM [29] for the purpose of experimental analysis. We use RBF kernel $K(u,v) = exp(\gamma||u-v||^2)$, where the parameter γ and penalty parameter C are chosen from $\{2^{-5}, 2^{-4}, \ldots, 2^4, 2^5\}$. We experimented with randomly selecting 80% of the data as training set and testing against remaining 20% of the data by selecting 10% at a time for 100 times. We repeat this process for 5 times and take the average of accuracy. We train the whole training set using the parameters that achieves the best validation rate and predict on the testing set.

TABLE II: Datasets description

Datasets	#Classes	Dimension	Size
Iris	3	4	150
Auto-mpg	3	7	398
SVMguide2	3	20	391
Vehicle	4	18	846
Pageblocks	5	10	5473
Glass	6	9	214
Satimage	6	36	6435
Segment	7	19	2310
Japanese-Vowel	9	12	9961
USPS	10	256	9298
Collins	15	23	1000
Yale	15	1024	165
Letter	26	16	20000
ORL	40	1024	400

In the first set of experiments, we carried out a comparative study of GP-SVM with the OAA-partition method with the same datasets used in [11]. We observe that our method gives almost similar accuracy as that of OAA-partition method. In order to explore further, we also compared the accuracy of the brute force method (i.e., exhaustive enumeration to find best binary partition) for the same datasets. The brute force method computes all possible combination at every stage and presumably determines the optimal partition. Table III reports the accuracy percentage of these three methods for the datasets used in [11]. We note that in majority of the cases the accuracy of all three methods are more or less the same. Thus, it can be inferred that for the chosen set of datasets all these methods return the best possible accuracy. We compare the testing time and training time of the three methods for these data sets for the sake of completeness.

In Table IV, time for training by different methods are analyzed. In this study, the time utilised for the construction of partition heirarchy is not considered and assumed to be a pre-processing step. This is standard in performance analysis of heirarchical classification. From Table IV, we observe that training time of GP-SVM is always better or equal as compared to OAA-partition. Though GP-SVM and Brute-force seems to be equal in training complexity (From Table IV). It is described in Section IV that determining the best binary partitioning by Brute-force method is not practical. However, GP-SVM outperforms OAA-partition in testing complexity(Table V). When the number of classes increases the advantage of GP-SVM becomes significant. It may be noted from the Tables (III-V) that Brute-force method could not solve the problem

TABLE III: Accuracy(%) comparison for the datasets used in [11]. For first six datasets, the accuracy is the same. There is slight improvement of accuracy for *Satimage, Segment, Yale* and *Letter* datasets. * denotes that the program did not terminate in two days.

Datasets	OAA-partition	Brute-force	GP-SVM
Iris	97.74	97.74	97.74
Auto-mpg	81.68	81.68	81.68
Svmguide2	84.00	84.00	84.00
Vehicle	84.62	84.62	84.62
Pageblocks	93.96	93.96	93.96
Glass	74.50	74.50	74.50
Satimage	91.14	91.52	91.52
Segment	96.17	96.34	96.34
Yale	78.07	*	78.27
Letter	94.83	*	94.99
ORL	96.20	*	95.95

in a reasonable time for the datasets with number of classes 15 or more.

TABLE IV: Training complexity in terms of time in seconds. * denotes that the program did not terminate in two days.

Dataset	OAA-partition	Brute-force	GP-SVM
Iris	0.03	0.03	0.03
Auto-mpg	0.06	0.05	0.06
Svmguide2	0.1	0.1	0.1
Vehicle	0.28	0.27	0.27
Pageblocks	3.05	2.99	3
Glass	0.1	0.09	0.09
Satimage	4.73	5.05	4.09
Segment	0.51	0.51	0.49
Yale	0.16	*	0.15
Letter	68.39	*	67.37
ORL	0.55	*	0.46

TABLE V: Classification complexity in terms of time in seconds which is proportional to number of SVMs invoked during testing. Significant gain in testing time can be noticed for datasets with large number of classes. * denotes that the program did not terminate in two days.

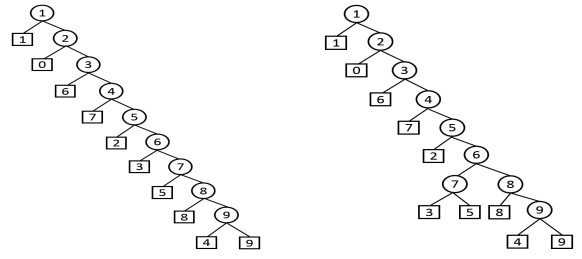
Dataset	OAA-partition	Brute-force	GP-SVM
Iris	0.28	0.28	0.28
Auto-mpg	0.60	0.60	0.60
Svmguide2	0.74	0.73	0.73
Vehicle	1.95	1.95	1.94
Pageblocks	29.89	29.86	29.85
Glass	0.74	0.74	0.74
Satimage	34.71	32.29	32.33
Segment	10.13	9.24	9.24
Yale	2.10	*	1.91
Letter	625.10	*	592.95
ORL	14.23	*	9.32

From the first set of experiments, we conclude that the advantages of different partitioning, in general and greedy partitioning in particular is not evident when the multiclass classification problem is for fewer number of classes. On the other hand, efficiency of a multiclass classifier can actually be tested for problems with large number of classes. In the second set of experiments we choose additional datasets where the number of classes are around 9 or more. Table VI summarizes the accuracy and depth of the tree for 3 datasets. It can be recalled that the testing time is proportional to the depth of the tree. We observe that for *Collins* dataset the depth of the tree resulting out of Greedy partitioning is 6 whereas the same for OAA-partitioning is 15. For classifying an unknown pattern, GP-SVM requires 6 SVMs at the worst case whereas OAA-partition requires 15 SVMs. This is a dramatic improvement of GP-SVM due to greedy partitioning. Not only the complexity of classification, the improvement is noticeable in terms of classification accuracy too. The average performance of GP-SVM in terms of accuracy of classification is 95.07% as against 91.1% for OAA-partition.

TABLE VI: Accuracy(%) comparison

Dataset	Accuracy		Height of the tree	
	OAA-partition	GP-SVM	OAA-partition	GP-SVM
Japanese-Vowel	98.72	98.73	8	7
USPS	96.86	97.18	9	8
Collins	91.1	95.07	15	6

Another measure of efficiency of a multiclass hierarchical SVM classifier is to examine the size of the training set of individual SVMs. It is appropriate to consider average size of the training set as a measure for comparison. It may be noted that for the toy example of 1207 data with 7 classes, the average size of the training sets for SVMs in OAA-partition is 620.96 whereas SVMs in GP-SVM require training sets of size 475.73 in average. Similarly, for Collins dataset, average size of training sets for OAA-partition is 499.31 whereas that of GP-SVM is 285.03. Thus, it is concluded that GP-SVM requires substantially smaller sized training sets than that of earlier method and yields higher accuracy of classification.



(a) OAA-partition method (b) GP-SVM method

Fig. 2: Tree resulting out of a. OAA-partition and b. Greedy partition for USPS dataset. Circles indicate non-leaf nodes of the tree with number for the nodes. Squares indicate leaf nodes with class labels.

Hierarchy Learning - In addition to higher efficiency, our method has advantage of learning the underlying hierarchy

of the set of classes. For instance, let us take USPS dataset which is the collection of handwritten digits for 0 to 9. The structure resulting out of OAA-partitioning does not depict any hierarchical relationship among classes (Figure 2a). On the other hand, the tree structure generated by greedy partitioning of GP-SVM carries very valuable information (Figure 2b). At node 6 of OAA-partition, class 3 (digit 3) is separated from the remaining classes- digits 5, 8, 4 and 9. On the other hand, the tree corresponding to greedy partition, at node 6, the classes 3 and 5 are separated from classes 8, 4 and 9. This indicates that handwritten digits 3 and 5 are similar and can be grouped together for deferred classification. Thus if we group the similarly-looking digits 3 and 5 together then one SVM classifies class 3&5 from class 4&8&9. Subsequently another SVM classifies digit 3 from digit 5. This is a more natural way of classification. Moreover, in this process, one learns the similarity between these two classes. Similar phenomenon is observed for classes 4 and 9.

VI. Conclusion and Future Work

In this paper we propose an efficient Binary Tree Structured SVM for multiclass classification. We show that the new classifier is better than the existing classifiers in many respects such as accuracy of classification, testing complexity, number of SVMs necessary for classification and average size of training sets for individual SVMs. We note that our method exhibits dramatic improvement for classification problem with large number of classes. Our experience with OCR odia character shows that GP-SVM is very suitable for correctly recognizing printed odia characters. We report this result in a separate paper. In addition to classification, the proposed technique also helps in understanding the class hierarchy of the underlying problem. In future, we propose to explore many other application areas with massive multiclass data.

References

[1] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems 12*, pp. 547–553, 1999.

[2] R. M. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

[3] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition," in *European Symposium on Artificial Neural Networks*, pp. 219–224, 1999.

[4] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[5] Y. Lee, Y. Lin, and G. Wahba, "Multicategory support vector machines, theory and application to the classification of microarray data and satellite radiance data," *Journal of the American Statistical Association*, vol. 99, pp. 67–81, 2004.

[6] Y. Liu and X. Shen, "Multicategory psi-learning," *Journal of the American Statistical Association*, vol. 101, no. 474, pp. 500–509, 2006.

[7] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," in *Neurocomputing*, pp. 41–50, Springer, 1990.

[8] S.-H. Park and J. Fürnkranz, "Efficient pairwise classification," in *European Conference on Machine Learning*, pp. 658–665, 2007.

[9] F. Schwenker, "Hierarchical support vector machines for multi-class pattern recognition," in *International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies*, pp. 561–565, 2000.

[10] B. Fei and J. Liu, "Binary tree of SVM: a new fast multiclass training and classification algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 696–704, 2006.

[11] X. Yang, Q. Yu, L. He, and T. Guo, "The one-against-all partition based binary tree support vector machine algorithms for multi-class classification," *Neurocomputing*, vol. 113, pp. 1–7, 2013.

[12] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[13] V. Vapnik, *Statistical learning theory*. Wiley, 1998.

[14] M. A. Kumar and M. Gopal, "Reduced one-against-all method for multiclass SVM classification," *Expert Syst. Appl.*, vol. 38, no. 11, pp. 14238–14248, 2011.

[15] J. H. Friedman, "Another approach to polychotomous classification," tech. rep., Department of Statistics, Stanford University, 1996.

[16] C. Freeman, D. Kulic, and O. Basir, "Feature-selected tree-based classification," *IEEE T. Cybernetics*, vol. 43, no. 6, pp. 1990–2004, 2013.

[17] V. Vural and J. G. Dy, "A hierarchical method for multi-class support vector machines," in *Machine Learning, Proceedings of the Twenty-first International Conference*, 2004.

[18] H. Lei and V. Govindaraju, "Half-against-half multi-class support vector machines," in *Multiple Classifier Systems, 6th International Workshop*, pp. 156–164, 2005.

[19] S. Cheong, S. H. Oh, and S.-Y. Lee, "Support vector machines with binary tree architecture for multi-class classification," *Neural Information Processing - Letters and Reviews*, vol. 2, no. 3, pp. 47–51, 2004.

[20] Y. Chen, M. M. Crawford, and J. Ghosh, "Integrating support vector machines in a hierarchical output space decomposition framework," in *IEEE International Geoscience and Remote Sensing Symposium*, pp. 949–952, 2004.

[21] S. Liu, H. Yi, L.-T. Chia, and D. Rajan, "Adaptive hierarchical multi-class svm classifier for texture-based image classification.," in *ICME*, pp. 1190–1193, 2005.

[22] X. Yuan, W. Lai, T. Mei, X. Hua, X. Wu, and S. Li, "Automatic video genre categorization using hierarchical SVM," in *Proceedings of the International Conference on Image Processing*, pp. 2905–2908, 2006.

[23] A. C. Lorena and A. C. P. L. F. de Carvalho, "Minimum spanning trees in hierarchical multiclass support vector machines generation," in *International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp. 422–431, 2005.

[24] G. Madzarov, D. Gjorgjevikj, and I. Chorbev, "A multi-class SVM classifier utilizing binary decision tree," *Informatica*, vol. 33, no. 2, pp. 225–233, 2009.

[25] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Advances in Neural Information Processing Systems*, pp. 163–171, 2010.

[26] Z. Liu, W. Shi, Q. Qin, X. Li, and D. Xie, "Hierarchical support vector machines," in *IEEE International Geoscience & Remote Sensing Symposium*, p. 4, 2005.

[27] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *IEEE International Conference on Computer Vision*, pp. 2072–2079, 2011.

[28] D. A. Cohen and E. A. Fernández, "SVMTOCP: A binary tree base SVM approach through optimal multi-class binarization," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 472–478, 2012.

[29] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.