GP-SVM: Tree Structured Multiclass SVM with Greedy Partitioning

Sandeep Kumar Sahu, Arun K. Pujari
Venkateswara Rao Kagita
Vikas Kumar, Vineet Padmanabhan

School of computers and information sciences
University of Hyderabad, Hyderabad

Overview of Presentation I

Multiclass SVM

- SVM is essentially for 2-class classification
- Many approaches to handle multiple classes
  - One-against-one (OAO-SVM)
  - One-against-all (OAA-SVM)
  - Direct Acyclic Graph (DAG-SVM)
  - Binary Tree structure (BT-SVM)
  - Multi-class optimization
- The approach that we use for solving multi class classification problem is Binary Tree Structure SVM. It takes advantage of both the efficient computation of the decision tree architecture and the high classification accuracy of SVMs.

Binary Tree SVM

- The hierarchy of binary decision subtasks using SVMs is designed with a clustering algorithm.
- It is natural to organize the SVMs as a binary tree.
- $K - 1$ SVMs are necessary during training of a K-class problem and only $log_2 K$ SVMs are required to classify an unknown sample. This leads to a dramatic improvement in recognition speed for problems with large K.

Table: 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	$K - 1$	$log_2(K)$ $\frac{K-1}{2}$	$\frac{N log_2(K)}{K-1}$ balanced $< \frac{N(K-2)}{K-1}$ unbalanced

Binarization I

- *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.
- Divide-by-2 method of Vural and Dy [9] and Half-Against-Half method of Lei and Govindaraju [4] are some of the first attempts in using hierarchical divisive clustering.
- Liu et al. [5] and Yuan et al. [11] employ K-Means clustering to group the classes into two subsets at every node.

Binarization II

- Lorena et al. [7] use the concept of Minimum Spanning Trees for binarization.
- Bengio et al. [1] use confusion matrix as an affinity matrix between classes and use spectral clustering.
- Cohen et.al. [2], each partition is solved by an optimized SVM and the one with the best performance is chosen at every node of the tree.
- Madzarov et al. [8] use farthest distance between class-centroids for divisive clustering.
- Liu et al. [6] use one class SVM to estimate pairwise distance between classes.

Binarization III

- Gao et.al [3], class hierarchies are constructed by postponing some decisions using uncertainty and resulting in higher recognition accuracy.

OAA-partition for binarization I

- Recently, Yang et al. [10], 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.
- 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.

OAA-partition for binarization II

- 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

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

where $l_i = |I_i|$ and $x_j^i \in I_i$.

OAA-partition for binarization III

- 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 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.

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

OAA-partition for binarization IV

- 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.
- Testing requires at most $K - 1$ SVMs to be revoked.

GP-SVM - Our Method I

- OAA-partition is very imbalance as it expands only one node at every stage.
- 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 $I_2 = I \setminus I_1$.
- We extend the OAA-partition to 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.

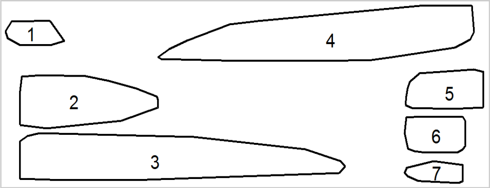
$$\mathbf{C}^s = \underset{\mathbf{C}' \subseteq \mathbf{C}}{argmax}\ PF(X', I^s \setminus X')$$

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

GP-SVM - Our Method II

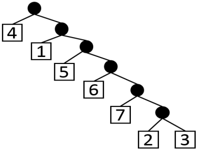
- The process is repeated recursively for both the subproblems $\mathbf{C}^s$ and $I^s \setminus \mathbf{C}^s$.

Figure: Example

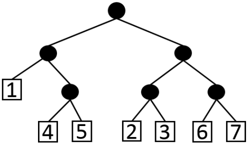


(a) Boundaries of seven classes

GP-SVM - Our Method III



(b) OAA-partition



(c) Exhaustive enumeration

- 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.
- Determining the best binary partition by exhaustive enumeration is not practical.

GP-SVM - Our Method IV

- Can there be an efficient method to compute true binary-tree structure using partition function as the separating measure?

GP-SVM - Our Method IV

- Can there be an efficient method to compute true binary-tree structure using partition function as the separating measure?
- If so, will it result in more accurate classifier?

GP-SVM - Our Method IV

- Can there be an efficient method to compute true binary-tree structure using partition function as the separating measure?
- If so, will it result in more accurate classifier?
- Can it handle massively large number of classes?

GP-SVM - Our Method IV

- Can there be an efficient method to compute true binary-tree structure using partition function as the separating measure?
- If so, will it result in more accurate classifier?
- Can it handle massively large number of classes?

GP-SVM

- We propose a greedy heuristic to determine the best binary partition which improves the classification accuracy and demands less computational overhead.
- 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.
- 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.

GP-SVM

- 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.

Example :

- We start with $\mathbf{C} = \{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_7}\}$.

GP-SVM

- 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.

Example :

- 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.

GP-SVM

- 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.

Example :

- 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 $I_2^t = \mathbf{C} \setminus \mathbf{I_1^t}$.

GP-SVM

- 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.

Example :

- 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 $I_2^t = \mathbf{C} \setminus \mathbf{I_1^t}$.
- $s = 2$
  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 $\{C_4, C_7\}$.

GP-SVM

- 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.

Example :

- 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 $I_2^t = \mathbf{C} \setminus \mathbf{I_1^t}$.
- $s = 2$
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 $\{C_4, C_7\}$.

GP-SVM

- 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 $X_t = X_1$.

GP-SVM

- 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.

GP-SVM

- 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 $\mathbf{C^s} = \{\mathbf{C_1}, \mathbf{C_4}, \mathbf{C_5}\}$.

GP-SVM

- 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 $\mathbf{C^s = \{C_1, C_4, C_5\}}$.
- We repeat the same procedure for the next stage by considering two subproblems and the partition.

GP-SVM

- 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 $\mathbf{C^s = \{C_1, C_4, C_5\}}$.
- We repeat the same procedure for the next stage by considering two subproblems and the partition.
- 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.

GP-SVM

- 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 $\mathbf{C^s = \{C_1, C_4, C_5\}}$.
- We repeat the same procedure for the next stage by considering two subproblems and the partition.
- 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.

Empirical Analysis I

Table: 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

Empirical Analysis II

Table: Accuracy(%) comparison for the datasets used in Yang et.al. [10]. 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

Empirical Analysis III

Table: 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	0.27	0.29	0.28
Pageblocks	3.93	5.67	4.92
Glass	0.1	0.12	0.1
Satimage	8.39	17.94	11.38
Segment	0.94	2.38	1.12
Yale	0.75	*	1.11
Letter	407.88	*	675.89
ORL	21.54	*	29.22

Empirical Analysis IV

Table: 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

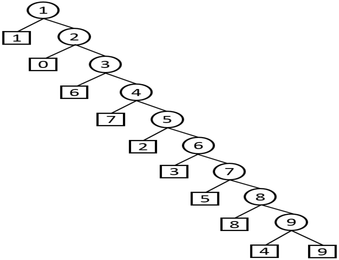
Empirical Analysis V

Table: Accuracy(%) comparison

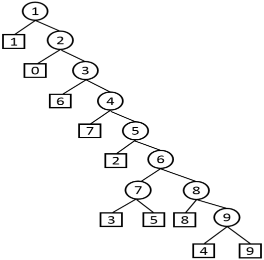
Dataset	**Accuracy**		**Height of the tree**	
	OAA-partition	**GP-SVM**	**OAA-partition**	**GP-SVM**
Japanese-Vowel	98.72	98.73	8	7
USPS	96.86	97.18	9	8
Collins	91.1	95.07	15	6

- Advantage of learning the underlying hierarchy of the set of classes.

Empirical Analysis VI



(d) OAA-partition method



(e) GP-SVM method

Figure: 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.

Conclusion and Future Work

- 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.
- 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.

Bibliography I

Samy Bengio, Jason Weston, and David Grangier.
Label embedding trees for large multi-class tasks.
In *Advances in Neural Information Processing Systems*, pages 163–171, 2010.

Diego Arab Cohen and Elmer Andrés Fernández.
SVMTOCP: A binary tree base SVM approach through optimal multi-class binarization.
In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 472–478, 2012.

Tianshi Gao and Daphne Koller.
Discriminative learning of relaxed hierarchy for large-scale visual recognition.
In *IEEE International Conference on Computer Vision*, pages 2072–2079, 2011.

Hansheng Lei and Venu Govindaraju.
Half-against-half multi-class support vector machines.
In *Multiple Classifier Systems, 6th International Workshop*, pages 156–164, 2005.

Song Liu, Haoran Yi, Liang-Tien Chia, and Deepu Rajan.
Adaptive hierarchical multi-class svm classifier for texture-based image classification.
In *ICME*, pages 1190–1193, 2005.

Zhigang Liu, Wenzhong Shi, Qianqing Qin, Xiaowen Li, and Donghui Xie.
Hierarchical support vector machines.
In *IEEE International Geoscience & Remote Sensing Symposium*, page 4, 2005.

Bibliography II

Ana Carolina Lorena and André Carlos Ponce Leon Ferreira 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*, pages 422–431, 2005.

Gjorgji Madzarov, Dejan Gjorgjevikj, and Ivan Chorbev.
A multi-class SVM classifier utilizing binary decision tree.
Informatica, 33(2):225–233, 2009.

Volkan Vural and Jennifer G. Dy.
A hierarchical method for multi-class support vector machines.
In *Machine Learning, Proceedings of the Twenty-first International Conference*, 2004.

Xiaowei Yang, Qiaozhen Yu, Lifang He, and Tengjiao Guo.
The one-against-all partition based binary tree support vector machine algorithms for multi-class classification.
Neurocomputing, 113:1–7, 2013.

Xun Yuan, Wei Lai, Tao Mei, Xian-Sheng Hua, Xiuqing Wu, and Shipeng Li.
Automatic video genre categorization using hierarchical SVM.
In *Proceedings of the International Conference on Image Processing*, pages 2905–2908, 2006.