

# Unbalanced Decision Trees for Multi-class Classification

A.Ramanan, S.Suppharangsarn, and M.Niranjan

Department of Computer Science, Faculty of Engineering, University of Sheffield, UK  
{A.Ramanan, Somjet, M.Niranjan}@dcs.shef.ac.uk

**Abstract** – In this paper we propose a new learning architecture that we call *Unbalanced Decision Tree (UDT)*, attempting to improve existing methods based on Directed Acyclic Graph (DAG) [1] and One-versus-All (OVA) [2] approaches to multi-class pattern classification tasks. Several standard techniques, namely One-versus-One (OVO) [3], OVA, and DAG, are compared against UDT by some benchmark datasets from the University of California, Irvine (UCI) repository of machine learning databases [4]. Our experiments indicate that UDT is faster in testing compared to DAG, while maintaining accuracy comparable to those standard algorithms tested. This new learning architecture UDT is general, and could be applied to any classification task in machine learning in which there are natural groupings among the patterns.

## I. INTRODUCTION

Classification is one of the standards of machine learning task. Many techniques such as Fisher's Linear Discriminant Analysis (LDA), Naïve Bayes, Perceptron, Neural Networks (NNs), Hidden Markov Models (HMMs), and Support Vector Machines (SVMs) are employed in various classification tasks. In general SVMs outperform other classifiers in its generalization performance [5]. SVMs were originally developed for solving binary classification problems [2] [6], but binary SVMs have also been extended to solve the problem of multi-class pattern classification which is still an ongoing research issue. There are three standard techniques frequently employed by SVMs to tackle multi-class problems, namely One-versus-One (OVO), One-versus-All (OVA), and Directed Acyclic Graph (DAG). For a  $k$  class problem, the OVO constructs  $k(k-1)/2$  binary classifiers, whereas OVA constructs  $k$  binary classifiers, and DAG implements an OVO based method arranging  $k(k-1)/2$  binary classifiers in a tree structure. However these standard techniques suffer from having a long evaluation time. For example, DAG needs to proceed until a leaf node is reached to make a decision on any input pattern. This problem becomes worse when  $k$  becomes large. In this paper, we propose a new learning architecture Unbalanced Decision Tree (UDT) to relieve the problem of excessive testing time while maintaining accuracy comparable to those standard techniques.

The remainder of this paper is organized as follows: The concept of multi-class classification and SVMs are described briefly in the rest of this section. More details about the standard techniques OVO, OVA, and DAG are

presented in section II and the new learning architecture UDT is described in section III. Section IV shows our experiments and results. Finally in Section V we give a discussion and present conclusions of our work.

## A Multi-class Classification

In multi-class classification each training point belongs to exactly one of the different classes. The goal is to construct a function which, given a new data point, will correctly predict the class to which the new data point belongs. Multi-class classification algorithms fall into two broad categories: the first type directly deals with multiple values in the target field, the second type breaks down the multi-class problem into a collection of binary class sub-problems and then combines them to make a full multi-class prediction. More generally, the second type contains a set of binary SVMs.

## B Support Vector Machines

SVMs are a supervised learning technique based on a statistical learning theory that can be used for pattern classification and regression. For the pattern classification case, SVMs have been used for isolated handwritten digit recognition [6], speaker identification [7], scene image classification [8], and pattern detection [9].

In pattern classification suppose we are given a set of  $l$  training points of the form:

$$(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l) \in \mathfrak{R}^n \times \{+1, -1\}$$

where  $x_i$  is an  $n$ -dimensional vector and  $y_i$  are their labels such that:

$$y_i = \begin{cases} +1: & \text{if the vector is classified to class } +1 \\ -1: & \text{if the vector is classified to class } -1 \end{cases}$$

We thus try to find a classification boundary function  $f(x) = y$  that not only correctly classifies the input patterns in the training data but also correctly classifies the unseen patterns.

The classification boundary  $f(x) = 0$ , is a *hyperplane* defined by its normal vector  $\mathbf{w}$ , which basically divides the input space into the class +1 vectors on one side and the class -1 vectors on other side. Then there exists  $f(x)$  such that

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \mathbf{w} \in \mathfrak{R}^n \text{ and } b \in \mathfrak{R}, \quad (1)$$

subject to

$$y_i f(x_i) \geq 1 \text{ for } i=1, 2, \dots, n. \quad (2)$$

The *optimal hyperplane* is defined by maximizing the distance between the hyperplane and the data points closest to the hyperplane (called *support vectors*). Then we need to maximize the margin  $\gamma = 2/\|\mathbf{w}\|$  or minimise  $\|\mathbf{w}\|$  subject to constraint (2). This is a quadratic programming (QP) optimization problem that can be expressed as:

$$\underset{w,b}{\text{minimise}} \quad \frac{1}{2} \|\mathbf{w}\|^2 \quad (3)$$

In practice, datasets are often not linearly separable in the input space. To deal with this situation slack variables ( $\xi_i$ ) are introduced [6] into (4), where  $C$  is the parameter that determines the tradeoff between the maximization of the margin and minimization of the classification error. The problem now becomes:

$$\underset{w,b}{\text{minimise}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \quad (4)$$

subject to

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall_i \quad (5)$$

The solution to the above optimization problem has the form:

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b = \sum_{i=1}^l c_i \phi(x_i) \cdot \phi(\mathbf{x}) + b \quad (6)$$

where  $\phi(\cdot)$  is the mapping function that transforms the vectors in input space to feature space. The dot product in (6) can be computed without explicitly mapping the points into feature space by using a kernel function (some example kernels are listed in Table I, where  $\gamma$ ,  $r$  and  $d$  are kernel parameters), which can be defined as the dot product of two points in the feature space:

$$K(x_i, x_j) + b \equiv \phi(x_i) \cdot \phi(x_j) \quad (7)$$

Thus the solution to the optimization problem has the form:

$$f(x) = \sum_{i=1}^l c_i K(x_i, x_j) + b \quad (8)$$

where most of the coefficients  $c_i$  are zero except for the coefficients of support vectors. Generally,  $K(x_i, x_j)$  satisfies the Mercer's Theorem [6] but the Sigmoid kernel does not satisfy the Mercer condition on all  $\gamma$  and  $r$  [10].

TABLE I  
EXAMPLES OF KERNELS

Linear	$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \cdot \mathbf{y}$
Polynomial	$K(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^T \cdot \mathbf{y} + r)^d, \quad \gamma > 0$
Radial Basis Function	$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \ \mathbf{x} - \mathbf{y}\ ^2), \quad \gamma > 0$
Sigmoid	$K(\mathbf{x}, \mathbf{y}) = \tanh(\gamma \mathbf{x}^T \cdot \mathbf{y} + r)$ for some $\gamma > 0$ and $r < 0$

## II. ONE-VERSUS-ONE, ONE-VERSUS-ALL, AND DIRECTED ACYCLIC GRAPH METHODS

OVO method is implemented using a ‘‘Max-Wins’’ voting strategy [11]. This method constructs one binary classifier for every pair of distinct classes, in total it constructs  $k(k-1)/2$  binary classifiers, where  $k$  is the number of classes. The binary classifier  $C_{ij}$  is trained with examples from the  $i^{\text{th}}$  class and the  $j^{\text{th}}$  class only, where examples from class  $i$  take positive labels (typically +1) while examples from class  $j$  take negative labels (typically -1). For an example  $x$ , if classifier  $C_{ij}$  predicts  $x$  is in class  $i$ , then the vote for class  $i$  is increased by one. Otherwise, the vote for class  $j$  is increased by one. The Max-Wins strategy then assigns  $x$  to the class receiving the highest voting score.

OVA method is implemented using a ‘‘Winner-Takes-All’’ strategy [11]. It constructs  $k$  binary classifier models where  $k$  is the number of classes. The  $i^{\text{th}}$  binary classifier is trained with all the examples in the  $i^{\text{th}}$  class with positive labels, and the examples from all other classes with negative labels. For an example  $x$ , the Winner-Takes-All strategy assigns it to the class with the highest classification boundary function value.

DAG SVMs are implemented using a ‘‘Leave-One-Out’’ strategy. The training phase of the DAG is the same as the OVO method, solving  $k(k-1)/2$  binary classifiers. In the testing phase it uses a rooted binary directed acyclic graph which has  $k(k-1)/2$  internal nodes and  $k$  leaves. Each node is a classifier  $C_{ij}$  from OVO. An example  $x$  is evaluated at the root node and then it moves either to the left or the right depending on the output value [1], as illustrated in Fig. 2(a).

## III. UNBALANCED DECISION TREE

UDT (see Fig. 2b) implements the OVA based concept at each decision node. Each decision node of UDT is an optimal classification model. The optimal model for each decision node is the OVA based classifier that yields the highest performance measure. Starting at the root node, one selected class is evaluated against the rest by the optimal model. Then the UDT proceeds to the next level by eliminating the selected class from the previous level of the decision tree. UDT terminates when it returns an output pattern at a level of the decision node, while DAG needs to proceed until a leaf node is reached. In contrast, we can say that UDT uses a ‘‘knock-out’’ strategy with at most  $(k-1)$  classifiers to make a decision on any input pattern and is an example of ‘vine’ structured testing strategy [12]. It will be a more challenging problem when  $k$  becomes very large.

We illustrate the construction of UDT during the training phase by using an example of vowel dataset of [13] with 4 classes (see Fig. 1). According to the UDT procedure 3-versus-All classifier is the optimal model at the root node as it is easily separated from the rest. In the next level the training data with class 3 labels are eliminated. At this level, 1-versus-All classifier is the optimal model.

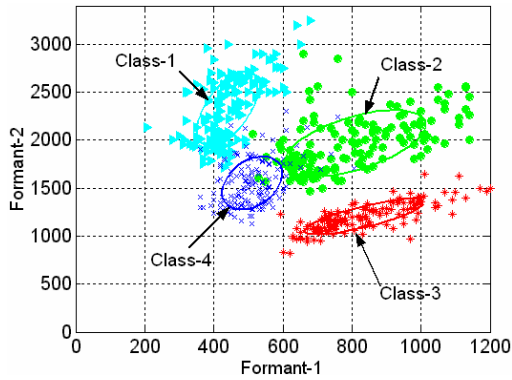


Fig. 1. Distribution of the first two formants of four classes selected from the vowel dataset [13].

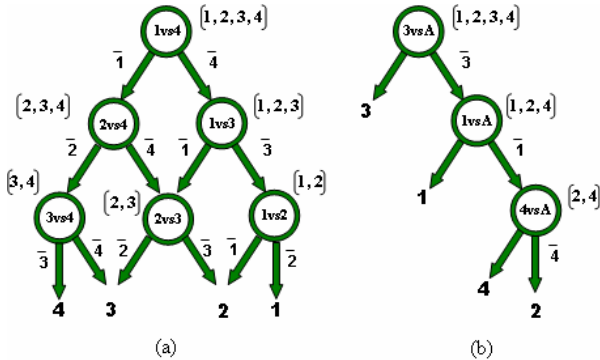


Fig. 2. DAG (a) and UDT (b) architectures and the classification problems at each node for finding the best class out of four classes indicated in Fig. 1. The equivalent list state for each node is shown next to that node.

At the leaf node the remaining training data with class 1 labels are eliminated and 4-versus-All classifier becomes the optimal model.

#### IV. EXPERIMENTS AND RESULTS

The experimental setup was such that, for the larger dataset *letter* we used 25% for testing, and a reduced training set was used by training only on 70% of the training set and validating on the other 30% of the training set. For smaller datasets a 10-fold cross validation was carried out on the datasets listed in Table II.

All the training data were scaled to be in  $[-1, 1]$ , then test data were adjusted using the same linear transformation. For each dataset, we considered Linear and Radial Basis Function (RBF) kernels in the experiment. Kernel parameters for the standard techniques were taken from [14]. For UDT, an initial experiment was performed to determine the optimal parameter(s) for each kernel type with a range of values of the cost parameters ( $C$ ) for the linear kernel model, and a set of combinations of parameters ( $C, \gamma$ ) for the RBF kernel model. Experiments were carried out using scripts in MATLAB embedding the SVM<sup>light</sup> toolkit [15]. We present the optimal parameter  $C$  for linear kernel and the corresponding accuracy rates in

TABLE II  
DATASET STATISTICS

Dataset	# data	#attributes	#classes
iris	150	4	3
wine	178	13	3
glass	214	9	6
vowel	528	10	11
vehicle	846	18	4
segment	2310	19	7
letter	20000	16	26

Table III, the optimal parameters ( $C, \gamma$ ) for RBF kernel and the corresponding accuracy rates in Table V. Also we present the testing time for solving the optimal model, averaged over the 10-fold cross validation runs in Table IV and Table VI respectively.

#### V. DISCUSSION AND CONCLUSIONS

With regard to training time, UDT needed to train more classifiers than any of the standard techniques so training time was longer. However, we may conclude by the results obtained that UDT is faster in testing compared to DAG, while maintaining accuracy comparable to those standard algorithms tested. Also we observe that in most cases the overall accuracy produced is much better when using RBF kernel.

Unfavourable classification results are due to the imbalanced datasets at decision nodes. One of the main reasons for the less performance is that the decision boundary between one ‘true’ class (typically +1) and its complementary combined ‘others’ class (typically -1) cannot be drawn precisely, due to the majority of data points in the ‘others’ class. We tested this effect by considering the classes 2 and 4 from the vowel dataset (see Fig. 1) and reducing the number of data points in class 2.

TABLE III  
A COMPARISON USING THE LINEAR KERNEL

Dataset	OVO		OVA		DAG		UDT
	$C$	rate	$C$	rate	$C$	rate	
iris	$2^4$	96.00	$2^{12}$	95.33	$2^8$	96.67	96.00
wine	$2^{-2}$	98.33	$2^2$	98.33	$2^{-2}$	98.33	97.22
glass	$2^8$	64.11	$2^5$	61.56	$2^4$	65.54	65.22
vowel	$2^5$	85.43	$2^{11}$	48.83	$2^6$	81.08	73.68
vehicle	$2^5$	80.16	$2^{12}$	74.37	$2^5$	80.87	77.69
segment	$2^{12}$	95.152	$2^{12}$	90.952	$2^{11}$	95.801	95.671
letter	$2^2$	84.760	$2^0$	59.060	$2^4$	83.660	74.800

TABLE IV  
A COMPARISON OF TESTING TIME (IN SECONDS) USING THE LINEAR KERNEL

Dataset	OVO	OVA	DAG	UDT
iris	0.06	0.06	0.46	0.40
wine	0.05	0.05	0.47	0.39
glass	0.32	0.13	1.50	1.26
vowel	1.20	0.33	7.44	4.29
vehicle	0.20	0.22	4.98	3.35
segment	0.94	0.47	21.30	15.39
letter	143.77	25.21	3682.02	1969.65

TABLE V  
A COMPARISON USING THE RBF KERNEL

Dataset	OVO		OVA		DAG		UDT rate
	$(C, \gamma)$	rate	$(C, \gamma)$	rate	$(C, \gamma)$	rate	
iris	$(2^{12}, 2^{-9})$	96.00	$(2^9, 2^{-3})$	96.00	$(2^{12}, 2^{-8})$	96.67	93.33
wine	$(2^7, 2^{-10})$	98.33	$(2^7, 2^{-6})$	98.33	$(2^6, 2^{-9})$	98.33	92.78
glass	$(2^{11}, 2^{-2})$	71.73	$(2^{11}, 2^{-2})$	71.33	$(2^{12}, 2^{-3})$	72.69	67.52
vowel	$(2^4, 2^0)$	99.62	$(2^4, 2^1)$	99.24	$(2^2, 2^2)$	99.43	97.55
vehicle	$(2^9, 2^{-3})$	85.46	$(2^{11}, 2^{-4})$	86.75	$(2^{11}, 2^{-5})$	86.17	84.14
segment	$(2^6, 2^0)$	96.970	$(2^7, 2^0)$	97.359	$(2^{11}, 2^{-3})$	96.970	97.143
letter	$(2^4, 2^2)$	97.570	$(2^2, 2^2)$	97.640	$(2^4, 2^2)$	97.590	96.360

TABLE VI  
A COMPARISON OF TESTING TIME (IN SECONDS) USING THE RBF KERNEL

Dataset	OVO	OVA	DAG	UDT
iris	0.05	0.05	0.43	0.36
wine	0.05	0.11	0.71	0.49
glass	0.48	0.13	1.47	1.36
vowel	1.26	0.32	7.35	5.45
vehicle	0.20	0.19	3.97	3.46
segment	1.29	0.57	19.34	14.60
letter	176.14	558.14	4392.56	2618.00

Corresponding class boundaries are presented in Fig 3(a) to Fig 3(c). The performance of the classifier at decision nodes could be improved by addressing the imbalance between classes in an appropriate manner [16].

Our long term interest is in addressing multi-class problems in which there are a large number of classes and a small number of data per class, such as those encountered in content based image retrieval (CBIR).

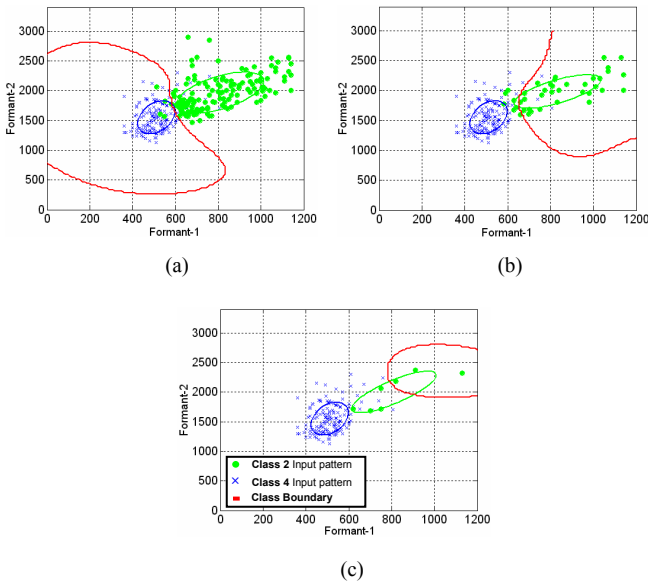


Fig. 3. Class 2 vs. 4 (a) balanced dataset (150:150) (b) slightly balanced dataset (50:150) (c) imbalanced dataset (10:150).

#### ACKNOWLEDGMENT

A.Ramanan's research studies are supported by the University of Jaffna, Sri Lanka, under the IRQUE Project funded by the World Bank, and the University of Sheffield, UK. S.Suppharangsarn's research studies are supported by Royal Thai Government.

#### REFERENCES

- [1] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large Margin DAGs for Multiclass Classification," *Advances in Neural Information Processing Systems*, vol. 12, pp. 547–553, 2000.
- [2] V. N. Vapnik, "The nature of statistical learning theory". New York, NY, USA: Springer-Verlag New York, Inc., 1995.
- [3] R. Debnath, N. Takahide, and H. Takahashi, "A decision based one-against-one method for multi-class support vector machine," *Pattern Anal. Appl.*, vol. 7, no. 2, pp. 164–175, 2004.
- [4] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, "UCI Repository of machine learning database," Irvine, CA, 1998.
- [5] C.J. Burgues, "A tutorial on Support Vector Machines for pattern recognition", *Knowledge Discovery and Data Mining*, vol. 2, no. 2, pp. 121-167, 1998.
- [6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [7] J. Salomon, S. King, and M. Osborne, "Frame-wise Phone Classification using Support Vector Machines," in *Proc. ICSLP*, Denver, Colorado, USA, pp. 2645-2648, 2002.
- [8] J. Ren, Y. Shen, S. Ma, and L. Guo, "Applying multi-class SVMs into scene image classification," in *IEA/AIE'2004: Proceedings of the 17th international conference on Innovations in applied artificial intelligence*, pp. 924–934, Springer Springer Verlag Inc, 2004.
- [9] H. Sahbi and D. Geman, "A Hierarchy of Support Vector Machines for Pattern Detection," *Journal of Machine Learning Research* 7, pp. 2087-2123, 2006.
- [10] H.-T. Lin and C.-J. Lin, "A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods", Technical report, National Taiwan University, 2003.
- [11] N. Cristianini and J. Shawe-Taylor, "An introduction to Support Vector Machines and other kernel-based learning methods". Cambridge University Press, March 2000.
- [12] G. Blanchard and D. Geman, "Hierarchical testing designs for pattern recognition", *The Annals of Statistics*, vol.33, no.3, pp. 1155-1202, 2005.
- [13] G. Peterson and H. Barney, "Control methods used in a study of the vowels", *J. Acoustical Society of America*, 24, 175-184, 1952.
- [14] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multi-class Support Vector Machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, March 2002.
- [15] T. Joachims, "SVM<sup>light</sup> - a software package for Support Vector Learning," 2004.
- [16] X. Hong, S. Chen, and C.J. Harris, "A Kernel-Based Two-Class Classifier for Imbalanced Datasets", *IEEE Transactions on Neural Networks*, vol.18, no.1, 2007.