On Boosting, *Tug of War*, and Lexicographic Programming

Shounak Datta, Sayak Nag, and Swagatam Das, *Senior Member, IEEE*

Abstract—Despite the large amount of research effort dedicated to adapting boosting for imbalanced classification, boosting methods are yet to be satisfactorily immune to class imbalance, especially for multi-class problems, due to the long-standing reliance on expensive cost set tuning. We show that the assignment of weights to the component classifiers of a boosted ensemble can be thought of as a game of *Tug of War* between the classes in the margin space. We then demonstrate how this insight can be used to attain a good compromise between the rare and abundant classes without having to resort to cost set tuning, which has long been the norm for imbalanced classification. The solution is based on a lexicographic linear programming framework which requires two stages. Initially, class-specific component weight combinations are found so as to minimize a hinge loss individually for each of the classes. Subsequently, the final component weights are assigned so that the maximum deviation from the class-specific minimum loss values (obtained in the previous stage) is minimized. Hence, the proposal is not only restricted to two-class situations, but is also readily applicable to multi-class problems. We also derive the dual formulation corresponding to the proposed framework. Experiments conducted on artificial and real-world imbalanced datasets as well as challenging applications such as hyperspectral image classification and ImageNet classification establish the efficacy of the proposal.

Index Terms—Boosting, Imbalanced classification, Lexicographic Linear Programming, Cost set tuning, Multi-Criterion Decision Making

1 Introduction

Boosting [1] is an ensemble learning technique that operates by repeatedly training a so-called weak classifier on reweighted versions of the basic dataset. The reweighting is done so that data instances misclassified in the previous round are assigned greater weights in the current round. The variants thus trained become the component classifiers of the ensemble having weightage proportional to their performance on the training data. Boosting is known to exhibit resistance to overfitting for noise-free datasets, owing to its ability to optimize the margin of the underlying weighted combination of weak learners [2].

However, boosting methods (and classification techniques in general) are unable to properly handle datasets characterized by class imbalance of data, i.e. when not all the classes in the dataset are equally represented in the training sample. Such imbalanced or uneven datasets often arise in critical real life applications such as medical diagnosis [3], fraud detection [4], etc. In fact, even the state-of-the-art deep-learning techniques which are used to solve complex computer vision applications suffer from class imbalance [5]. to Nikolaou et al. [6], in a rather exhaustive comparative study, observe that a significant amount of research effort has been aimed towards adapting boosting methods like AdaBoost [7] for such class imbalanced learning tasks. Despite the continued research efforts [8]–[14], boosting methods have yet to become sufficiently immune to class imbalance due to the following reasons:

- Most of the boosting variants proposed to handle class imbalance [6], [8], [11], [15] assume that the relative costs of misclassifying the two classes are known *a priori*. This is often not true and the set of relative costs that are most suitable for a particular dataset must be found by a costly parameter tuning regime.
- Moreover, most of the research efforts have been aimed at handling dichotomous or the so-called two-class imbalanced problems, with little attention being accorded to multi-class or polychotomous classification problems characterized by class imbalance. One of the reasons behind this is the need for parameter tuning, which becomes exponentially costlier for multi-class datasets.

Based on the generally accepted notion that the minimum margin is key to generalization performance, Grove and Schuurmans [16] presented an interesting variant of boosting, called LPAdaBoost, where the weights of the component classifiers are chosen by a linear program so as to maximize the minimum margin. LPAdaBoost has hitherto unexplored applicability to class imbalanced problems. Since the learner is overwhelmed by the abundance of the majority data instances, it is likely to classify most data into the majority class, resulting in high margin for the majority instances and low margin for the minority instances. Therefore, LPAdaBoost attempting to compensate for the margin imbalance, can effectively compensate for class imbalance. However, in the presence of outliers, data noise or label noise, the minimum margin is likely to correspond to corrupt instances, leading to a complete miscalibration of the component weights. Rätsch et al. [17] proposed the LPBoost algorithm which aims to solve the miscalibration issue by regularizing the noisy and outlier instances. A suitable cost of regularization must be specified for LPBoost to work well. Unfortunately, such an approach compromises the applicability to imbalanced problems as most

- *Shounak Datta (Email: shounak.jaduniv@gmail.com) and Swagatam Das (Email: swagatam.das@isical.ac.in) are with the Electronics and Communication Sciences Unit, Indian Statistical Institute, 203, B. T. Road, Kolkata-700 108, India.*
- *Sayak Nag was formerly with the Instrumentation & Electronics Engineering Department, Jadavpur University Salt Lake Campus, Salt Lake City , Block-LB, Plot No. 8, Sector - III, Kolkata - 700 098, India. Email: sayak.nag9@gmail.com.*
- *Corresponding author: Swagatam Das*

of the regularization would be from the minority class. Leskovec and Shawe-Taylor [18] proposed LPUBoost, attempting to solve this problem by using higher regularization cost for the minority instances. While this can solve the issue of uneven regularization, it also reinstates the long-standing issue of cost set selection. The best set of relative weights depends on a variety of factors such as the relative densities of the classes, the extent and structure of the overlap (if any) between the classes, the amount of noise/outliers, etc. and must be identified by a computationally expensive cost set tuning regime. Moreover, the issue of cost set tuning prevents the extension of such methods to other inherently imbalanced problems such as multi-class imbalance, online learning, single-class classification, etc.

1.1 Boosting as a game of *Tug of War*

Perhaps a better understanding of the effects of the component weights on the margin values can help us extend the elegant, cost-independent effectiveness of LPAdaBoost to noisy imbalanced problems.

Definition 1. *Let* $X = \{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, n; y_i \in \{-1, +1\}\}$ *be a given training dataset and let* $\{f_1, f_2, \cdots, f_T\}$ *be the set of component classifiers having corresponding set of component weights* $\{\alpha_1, \alpha_2, \cdots, \alpha_T\}$. *Then for a data point* $\mathbf{x}_i \in X$, *the margin* $\rho(\mathbf{x}_i)$ *is defined as*

$$\rho(\mathbf{x}_i) = y_i h(\mathbf{x}_i),$$

where

$$h(\mathbf{x}_i) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i),$$

and is hereafter referred as the "signed margin" of the point $\mathbf{x}_i$ *while* $f_t(\mathbf{x}_i)$ *denotes the output of the* t*-th classifier for* $\mathbf{x}_i$.

Let us consider the simple imbalanced dataset exhibited in Figure 1. It is seen that both the majority as well as the minority classes contain noisy instances. We run three rounds of AdaBoost on this dataset to obtain three component classifiers. The signed margin values achieved by the data instances for two arbitrary component weight combinations are shown in Figure 2. Ideally, each point in the positive class should have a signed margin value of +1 while all the negative points should have a signed margin value of -1 (i.e. all points should ideally have margin values of +1). An inspection of Figure 2 shows that some of the points from both the classes always attain the ideal margin value. These correspond to the data points which are correctly classified by all the component classifiers. However, the margin values attained by other points, which are correctly classified only by some of the components, depends on the choice of component weights. If the components which correctly classify most of the majority class instances (generally at the cost of the minority class instances) are assigned high weightage, most of the majority points will have high margin values while many of the minority points will have low margin values. Similarly, if high weightage is assigned to classifiers performing well on the minority points, the status quo will be reversed. Therefore, one can think of the problem of component weight assignment as a game of *Tug of War* between the classes in the signed margin space[1].

1. This game of *Tug of War* is analogous yet distinct from the zero-sum versions of [19], [20]. While the a decrease in the hinge loss of one class is likely to result in an increase in the loss for the other class, the net sum of changes in this game may not be zero.

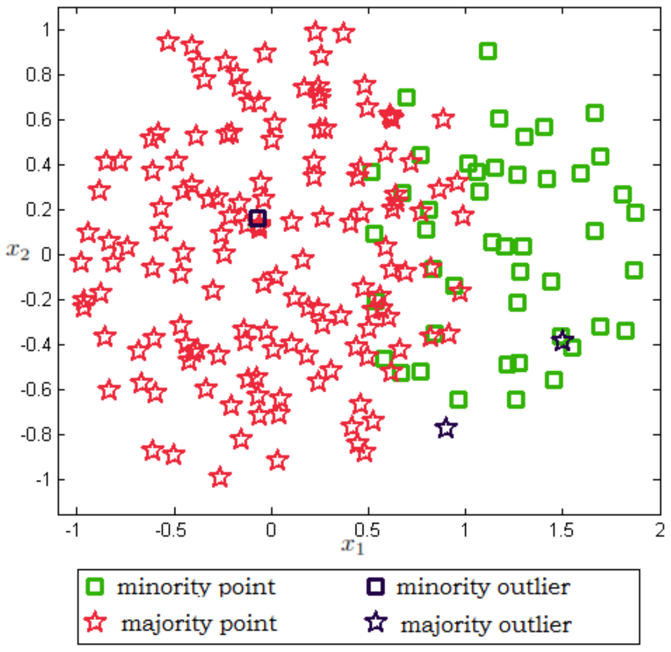


Fig. 1. Toy class imbalanced dataset with noisy instances in both classes.

Intuitively, if both classes are of equal importance or if no cost information is available, the best trade-off solution is to select the component weights so that similar fractions of data points are misclassified from both classes. However, in the presence of noise, the corrupt instances are added to the fractions of misclassification for the two classes, resulting in miscalibration. Such miscalibration leads to the failure of hard margin maximization [17] in the presence of noise. Therefore, there is a need to regularize the noisy and outlier instances by some means. Since the traditional method of using slack variables to achieve such regularization results in a need for cost set tuning [18], we are motivated to devise a new framework which will not only regularize the noisy and outlier instances but also strike a good compromise between the two-classes without resorting to cost set tuning.

The key idea is to solve the problem in two sequential steps. In the first step, we find the two (possibly different) sets of component weights which minimize the average hinge loss (on the difference between the actual and ideal margin values) for the two classes individually. The hinge loss is defined as follows:

Definition 2. *For a data instance* $\mathbf{x}_i \in X$, *the hinge loss on the difference* $\rho(\mathbf{x}_i) - 1$ *is defined as*

$$L_h(\mathbf{x}_i) = \begin{cases} 0 \textit{ if } \rho(\mathbf{x}_i) >= 1, \\ 1 - \rho(\mathbf{x}_i) \textit{ if } \rho(\mathbf{x}_i) < 1. \end{cases}$$

In other words, the hinge loss is the extent by which the margin of a point is worse than the ideal value. Since the noisy and outlier instances from a class are misclassified by most of the component classifiers, these instances will have high loss values even when the overall average loss for the class is minimized. The signed margin values, achieved by minimizing the average hinge loss over the data points in the majority and the minority classes, are displayed respectively in Figure 3. It can be seen that, in both the cases, the signed margin values thus achieved result in the regularization

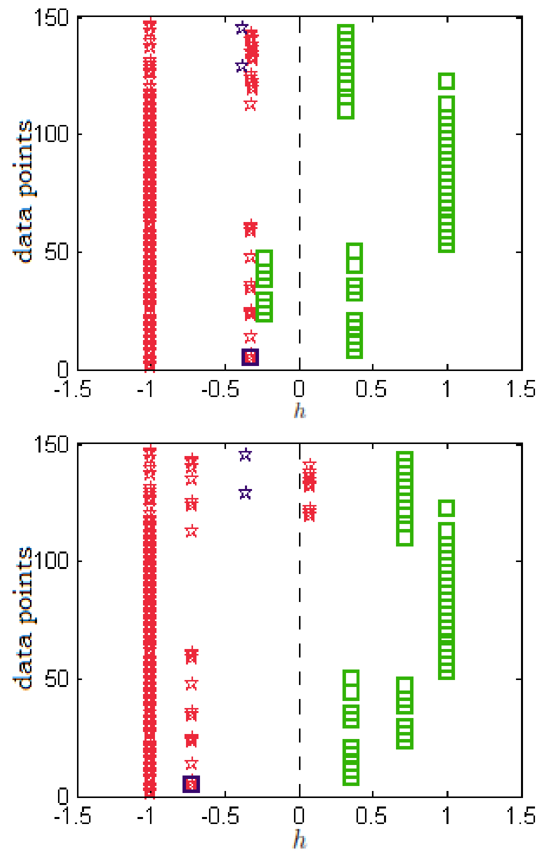


Fig. 2. Margin values obtained by the data for two arbitrary component weight combinations. Follow legends of Figure 1.

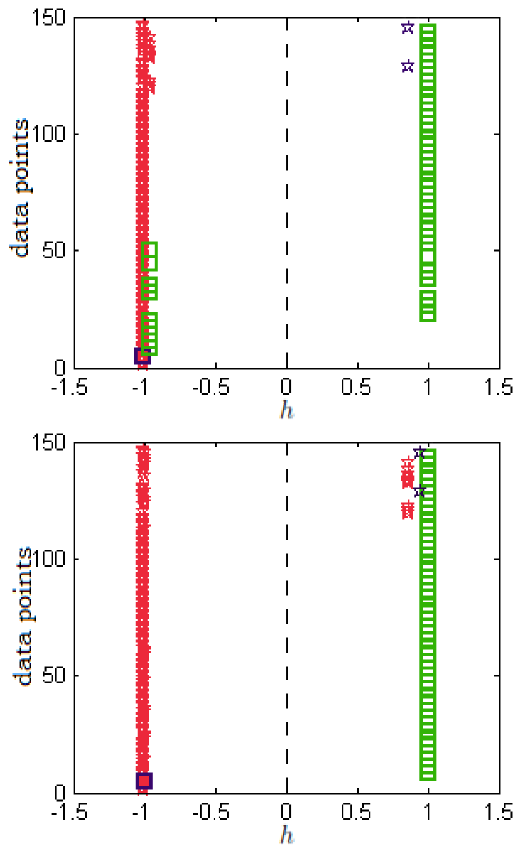


Fig. 3. Margin values obtained by minimizing class-specific hinge loss for the majority class (above) and the minority class (below). Follow legends of Figure 1.

of the outliers (in the sense that the outliers have worse signed margin and consequently higher loss).

Now, any set of component weights which seeks to achieve a compromise between the classes will result in an increase (beyond the minimum average hinge losses found in the first step) in the class-wise average losses for both the classes. Since the noisy and outlier instances already had high loss values, most of this increase in loss will be due to the misclassification of the non-outlier instances. Therefore, similar increase in the average hinge losses of the two classes will correspond to similar fraction of misclassification of non-outlier points for the two classes. Hence, in the second step, the final component weight set (that strikes a good balance between the two classes) can be found by minimizing the maximum increase in the class-wise average hinge losses. Based on these insights, we propose a Lexicographic Linear Programming (LLP) based approach (described in Section 1.2) to choose the optimal set of component classifier weights for boosting. The proposed approach is expected to be proficient at handling class imbalanced classification tasks, without having to partake in cost set tuning.

1.2 Contributions

Lexicographic programming techniques, such as goal programming, have long been used for multi-criteria decision analysis [21]. Class imbalanced classification can also be thought of as a multi-criteria decision making problem, since the classification accuracy on the majority as well as the minority classes must be simultaneously maximized (these two objectives are often contradictory and cannot be maximized together, resulting in the need for a suitable trade-off). In spite of this, to the best of our knowledge, the current article is the first application of lexicographic programming to the class imbalanced classification problem.

Definition 3. *We formally define an LLP as a lexicographic hierarchy of LPs (in the sense that the LPs in all the prior stages of the hierarchy must be solved before the LPs in the current stage can be solved). The j-th LP to be solved at the i-th stage is of the form*

$$\mathbf{v}^*_{ij} = \arg\min_{\mathbf{v}} L_{ij}(\mathbf{v}, \mathbf{v}^*_{i-1}), \tag{1}$$

$$\textit{s. t. } g_k(\mathbf{v}, \mathbf{v}^*_{i-1}) \leq 0 \; \forall k \in \{1, \cdots, \eta_i\} \tag{2}$$

$$\textit{and } h_l(\mathbf{v}, \mathbf{v}^*_{i-1}) = 0 \; \forall l \in \{1, \cdots, \nu_i\}, \tag{3}$$

where η_i and ν_i respectively are the number of inequality and equality constraints while L_{ij} is an appropriate loss function. The vector $\mathbf{v}^*_{i-1} = (\mathbf{v}^*_{(i-1)1}, \cdots, \mathbf{v}^*_{(i-1)\zeta_{i-1}}, \cdots, \mathbf{v}^*_{11}, \cdots, \mathbf{v}^*_{1\zeta_1})$ contains the optimal solutions to all LPs solved in all the preceeding stages with ζ_i denoting the number of LPs solved in the i-th stage.

We propose a two staged LLP scheme to choose the weightage of the component classifiers of a boosted ensemble. The proposed method is referred to as LexiBoost hereafter. The first stage of LexiBoost is concerned with solving a set of Linear Programs (LPs) (one for each class, which can be solved in parallel) to minimize the average class-wise hinge losses (see Definition 2) for each of the classes. Subsequently, the second stage solves another LP to find the set of component weights that minimizes the maximum of the class-wise deviations from the optimal average loss values found in the first stage.

The proposed method has the following advantages:

- The novel hinge loss based regularization method, unlike slack variable based regularization, does not require cost set tuning to achieve a good balance between the classes, thus addressing the long-standing issue of expensive cost set tuning for imbalanced data learning.
- Moreover, the proposed approach is readily applicable to multi-class or polychotomous learning tasks, which have as yet received limited attention in the class imbalanced learning literature.
- Even though we demonstrate the abilities of the proposal using the AdaBoost algorithm, the proposed philospohy can be applied to other ensemble learning techniques as well.

1.3 Organization

We introduce the reader to the popular AdaBoost algorithm as well as some of the existing LP based boosting schemes in Section 2. We then provide a detailed explanation of the proposed two staged LLP based LexiBoost framework in Section 3. The dual formulation resulting from the proposed LLP is presented in Section 3.3. The proposed framework is also generalized to multi-class classification problems in Section 3.4. Subsequently, experimental results are presented in Section 4. We conclude with some relevant discussion in Section 5.

2 Prior Work

In this section, we introduce the reader to some of the extant boosting techniques which are crucial to understanding the proposed improvement.

2.1 AdaBoost

Freund and Schapire [7] proposed the AdaBoost ensemble learning technique which is considered to be one of the top algorithms in the field of machine learning [22]. The basic idea is to generate diverse classifiers by assigning more weightage $D_{t+1}(i)$, in the $(t+1)$-th round, to the data points $\mathbf{x}_i$ which prove to be difficult in the t-th round. Algorithm 1 describes the complete method.

Algorithm 1: AdaBoost

```
Input: Dataset $X = \{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, n\}$.
Output: Final ensemble classifier
        $H(\mathbf{x}_i) = \text{sign}(\sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i))$.

Initialize $D_1(i) = 1/n \; \forall i = 1, 2, \cdots, n$.
for all $t = 1$ to $T$ do
    Train weak learner $f_t$ using distribution $D_t$.
    Calculate training error $\epsilon_t$ using distribution $D_t$.
    if $\epsilon_t$ >0.5 then
        break
    end if
    Choose $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$.
    Update $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i f_t(\mathbf{x}_i))}{Z_t}$
    $\forall i = 1, 2, \cdots, n$, where $Z_t$ is a normalization factor.
end for
```

2.2 Linear Programming based Boosting

LPAdaBoost: Despite the theoretical guarantees on the training performance of AdaBoost [7], Grove and Schuurmans [16] proposed the LPAdaBoost algorithm to maximize the minimum margin ρ, aiming to achieve better generalization performance. The primal LP posed by LPAdaBoost is of the form

$$\text{A}_1\colon (\boldsymbol{\alpha}^*, \rho^*) = \arg\max \rho, \tag{4}$$

$$\text{s. t. } y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i) \geq \rho \; \forall i \in \{1, \cdots, n\}, \tag{5}$$

$$\sum_{t=1}^{T} \alpha_t = 1, \tag{6}$$

$$\text{and } \boldsymbol{\alpha} \geq 0, \tag{7}$$

where $f_t(\mathbf{x}_i)$ is the classifier generated by the t-th round of AdaBoost.

Dual-LPAdaBoost: The dual to the primal LPAdaBoost formulation is

$$\text{A}_1'\colon (\boldsymbol{D}_{t+1}^*, s^*) = \arg\min s, \tag{8}$$

$$\text{s. t. } \sum_{i=1}^{n} D_{t+1}(i) y_i f_\tau'(\mathbf{x}_i) \leq s \; \forall \tau \in \{1, \cdots, t\}, \tag{9}$$

$$\sum_{i=1}^{n} D_{t+1}(i) = 1, \tag{10}$$

$$\text{and } \boldsymbol{D}_{t+1} \geq 0, \tag{11}$$

where $f_\tau'(\mathbf{x}_i)$ is the classifier generated in the τ-th round. The dual formulation corresponds to assigning the point-wise weights $D_{t+1}(i)$ such that the aggregate margin s is minimized. To put it simply, the dual attempts to find a $\boldsymbol{D}_{t+1}$ which assigns the greatest weightage to the points which prove to be the most difficult during rounds 1 through t. The Dual-LPAdaBoost algorithm consists of alternatingly solving the LPs A_1 and A_1' until the convergence criterion $s - \rho < 0$ is met or the maximum number of rounds T is reached.

LPBoost: Since the hard margin formulation of LPAdaBoost makes it sensitive to noise and outliers, Rätsch et al. [17] presented a soft margin variant called LPBoost which regularizes the noisy and outlier instances using slack variables ξ_i (corresponding to the data points $\mathbf{x}_i$), resulting in the following LP:

$$\text{A}_2\colon (\boldsymbol{\alpha}^*, \boldsymbol{\xi}^*, \rho^*) = \arg\min -\rho + D \sum_{i=1}^{n} \xi_i, \tag{12}$$

$$\text{s. t. } y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i) \geq \rho - \xi_i \; \forall i \in \{1, \cdots, n\}, \tag{13}$$

$$\sum_{t=1}^{T} \alpha_t = 1, \tag{14}$$

$$\boldsymbol{\alpha} \geq 0, \tag{15}$$

$$\text{and } \boldsymbol{\xi} \geq 0. \tag{16}$$

An appropriately high cost D must be assigned for the regularization of data points in order to achieve good performance. This parameter has to be generally selected by cross-validation on the training data.

DualLPBoost: The dual LP arising out of the LPBoost formulation is of the form

$$\mathrm{A}_2'\colon (\boldsymbol{D}_{t+1}^*, s^*) = \arg\min s, \tag{17}$$

$$\text{s. t.} \sum_{i=1}^{n} D_{t+1}(i) y_i f_\tau(\mathbf{x}_i) \leq s \;\forall \tau \in \{1, \cdots, t\}, \tag{18}$$

$$\sum_{i=1}^{n} D_{t+1}(i) = 1, \tag{19}$$

$$\text{and } 0 \leq \boldsymbol{D}_{t+1} \leq D, \tag{20}$$

giving rise to the Dual-LPBoost algorithm where A_2' is solved for a maximum of T rounds (until the convergence criterion $\sum_{i=1}^{n} D_t(i) y_i f_t(\mathbf{x}_i) \leq s$ is satisfied) with the Lagrangian multipliers of A_2' being chosen to be the component weights α_t.

LPUBoost: Leskovec and Shawe-Taylor [18] further adapted the LPBoost formulation to two-class imbalanced problems by introducing uneven costs for regularizing the two classes. The non-target (usually majority) class is assigned a regularization cost of D as in LPBoost, while the target (usually minority) class is assigned a higher regularization cost of βD $(\beta > 1)$. The resulting primal LP is

$$\mathrm{A}_3\colon (\boldsymbol{\alpha}^*, \boldsymbol{\xi}^*, \rho^*) = \arg\min -\rho + \beta D \sum_{i=1}^{n_1} \xi_i + D \sum_{i=n_1+1}^{n} \xi_i, \tag{21}$$

$$\text{s. t. } y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i) \geq \rho - \xi_i \;\forall i \in \{1, \cdots, n\}, \tag{22}$$

$$\sum_{t=1}^{T} \alpha_t = 1, \tag{23}$$

$$\text{and } (\boldsymbol{\alpha}, \boldsymbol{\xi}) \geq 0, \tag{24}$$

where n_1 denotes the number of points in the positive (target) class. Hence, the number of points in the negative (non-target) class is $n_2 = n - n_1$. Both the parameters D as well as β must be selected by expensive tuning on $\mathbb{R}^+ \times \mathbb{R}^+$ using cross-validation. Tuning the parameter β essentially corresponds to tuning the relative cost between the two classes, and is critical to achieving good performance.

Dual-LPUBoost: Like LPAdaBoost and LPBoost, LPUBoost also gives rise to a dual problem. The dual problem is of the form

$$\mathrm{A}_3'\colon (\boldsymbol{D}_{t+1}^*, s^*) = \arg\min s, \tag{25}$$

$$\text{s. t.} \sum_{i=1}^{n} D_{t+1}(i) y_i f_\tau(\mathbf{x}_i) \leq s \;\forall \tau \in \{1, \cdots, t\}, \tag{26}$$

$$\sum_{i=1}^{n} D_{t+1}(i) = 1, \tag{27}$$

$$0 \leq D_{t+1}(i) \leq \beta D \;\forall i \in \{1, \cdots, n_1\}, \tag{28}$$

$$\text{and } 0 \leq D_{t+1}(i) \leq D \;\forall i \in \{n_1 + 1, \cdots, n\}. \tag{29}$$

However, the Dual-LPUBoost algorithm solves a slightly modified from of the LP A_3' to accommodate for the drawbacks of simple cost set tuning. While the modification does seem to lend some robustness to the method (see Section 4), it also adds an additional tunable parameter D_{LB}, which determines the lower limit of $D_{t+1}(i)$ as a fraction of the corresponding upper limit. Thus, the changes pertain to the constraints (28) and (29), resulting in the modified constraints

$$\beta D \times D_{LB} \leq D_{t+1}(i) \leq \beta D \;\forall i \in \{1, \cdots, n_1\}, \tag{30}$$

$$\text{and } D \times D_{LB} \leq D_{t+1}(i) \leq D \;\forall i \in \{n_1 + 1, \cdots, n\}. \tag{31}$$

The termination criterion and the choice of component weights are identical to those of Dual-LPBoost.

3 Lexicographic Linear Programming based selection of component weights

Having acquainted the reader to the basic AdaBoost algorithm and the existing LP based improvements, we now elucidate the proposed LexiBoost algorithm which uses a two staged LLP. The two stages of LP involved in our proposed LLP framework are formally defined in Sections 3.1 and 3.2.

3.1 The first stage of LPs

As already mentioned in Section 1.2, the initial aim is to choose the component classifier weights so that the average hinge loss on the differences between the actual and ideal margins is minimized for the individual classes. Since the hinge loss is piece-wise linear, the minimization problems can be posed as LPs. Therefore, an LP P_j (j denotes the class in question) of the following form must be solved for each of the classes:

$$\mathrm{P}_j\colon (\boldsymbol{\alpha}_j, \boldsymbol{\lambda}_j^*) = \arg\min \frac{1}{n_j} \sum_{i=1}^{n_j} \lambda_i, \tag{32}$$

$$\text{s. t. } 1 - \rho(\mathbf{x}_i) \leq \lambda_i \;\forall i \in \{1, \cdots, n_j\}, \tag{33}$$

$$\sum_{t=1}^{T} \alpha_t = 1, \tag{34}$$

$$\boldsymbol{\alpha} \geq 0, \tag{35}$$

$$\text{and } \boldsymbol{\lambda} \geq 0, \tag{36}$$

where n_j is the number of points in the j-th class ($j \in \{1, \cdots, |\mathcal{C}|\}, \mathcal{C} = \{c_1, c_2\}$ for the two-class imbalanced problem being the set of classes), $\boldsymbol{\lambda}_j^*$ denotes the optimal set of λ_i values obtained by solving the LP P_j, and $\boldsymbol{\alpha}_j$ is the corresponding set of component weights.

3.2 The final LP

Since the noisy and outlier instances already have high loss values even when the overall class-wise losses are minimized, any further increase in the class-wise losses is likely to be due to regularization of non-outlier instances. Therefore, having obtained the optimal average hinge losses for each of the individual classes, the deviations from optimal average losses must be minimized to restrict the regularization (and hence misclassification) of non-noisy points. Hence, we finally solve another LP to find the α_t values which minimize the maximum of such deviations. It should be noted that unlike P_j, all the classes (and hence all the training data points) are considered together at this stage. The maximum deviation is denoted by χ (χ^* being the optimized value) and the final LP is referred to as Q. We now present the formulation of Q for obtaining the optimal set of component weights $\boldsymbol{\alpha}^*$ which are expected to strike a good balance between the classes. The formulation for the program is as follows:

$$\text{Q: } (\boldsymbol{\alpha}^*, \boldsymbol{\lambda}, \chi^*) = \arg\min \chi, \tag{37}$$

$$\text{s. t. } \frac{1}{n_j}\sum_{i\in c_j}\lambda_i - \frac{1}{n_j}\mathbf{e}^T\boldsymbol{\lambda}_j^* \le \chi \; \forall j \in \{1,\cdots,|\mathcal{C}|\}, \tag{38}$$

$$1-\rho(\mathbf{x}_i) \le \lambda_i \; \forall i \in \{1,\cdots,n\}, \tag{39}$$

$$\sum_{t=1}^{T}\alpha_t = 1, \tag{40}$$

$$\text{and } (\boldsymbol{\alpha}, \boldsymbol{\lambda}, \chi) \ge 0, \tag{41}$$

where $\mathbf{e}$ denotes a vector of ones, having appropriate length. Hence, the complete LexiBoost algorithm (in its primal formulation) is presented in Algorithm 2.

```
Algorithm 2: LexiBoost
Input: Dataset $X$ of labelled instances $\mathbf{x}_i$.
Output: Final ensemble classifier
        $H(\mathbf{x}_i) = \text{sign}(\sum_{t=1}^{T}\alpha_t^* f_t(\mathbf{x}_i))$.
Run AdaBoost on the dataset $X$ to obtain $f_t(\mathbf{x}_i)$.
for all $j \in \{1,\cdots,|\mathcal{C}|\}$ do
    Solve LP $P_j$ to obtain $\boldsymbol{\lambda}_j^*$.
end for
Solve LP Q to obtain $\boldsymbol{\alpha}^*$.
```

3.3 The dual to LexiBoost

Since the dual to boosting techniques are often of interest [23], in this section we present the dual formulation of the proposed LexiBoost algorithm. The dual LPs resulting from the LPs P_j are of the form

$$\text{P}'_j\text{: } (\boldsymbol{D}_{t+1}^*, s^*) = \text{maximize } \sum_{i=1}^{n_j} D_{t+1}(i) - s, \tag{42}$$

$$\text{s. t. } \sum_{i=1}^{n} D_{t+1}(i) y_i f_\tau(\mathbf{x}_i) \le s \; \forall \tau \in \{1,\cdots,t\}, \tag{43}$$

$$\sum_{i=1}^{n} D_{t+1}(i) = 1, \tag{44}$$

$$0 \le D_{t+1}(i) \le \frac{1}{n_j} \; \forall i \in c_j, \forall j \in \{1,\cdots,|\mathcal{C}|\}. \tag{45}$$

The interpretation is similar to that of LPAdaBoost in that the dual formulation attempts to assign higher weights to the points which have proved to be difficult in the previous rounds 1 through t while also maximizing the sum of weights. The constraints $0 \le D_{t+1}(i) \le \frac{1}{n_j} \; \forall i \in c_j, \forall j \in \{1,\cdots,|\mathcal{C}|\}$ ensure that higher weights are assigned to instances belonging to the minority class.

Additionally, the dual LP which follows from the final LP Q is as follows:

$$\text{Q}'\text{: } (\boldsymbol{D}_{t+1}^*, \boldsymbol{d}, s^*) = \text{max. } \sum_{i=1}^{n_j} D_{t+1}(i) - \sum_{j=1}^{|\mathcal{C}|}\frac{d_j}{n_j}\mathbf{e}^T\boldsymbol{\lambda}_j^* - s, \tag{46}$$

$$\text{s. t. } \sum_{i=1}^{n} D_{t+1}(i) y_i f_\tau(\mathbf{x}_i) \le s \; \forall \tau \in \{1,\cdots,t\}, \tag{47}$$

$$\sum_{i=1}^{n} D_{t+1}(i) = 1, \tag{48}$$

$$0 \le D_{t+1}(i) \le \frac{d_j}{n_j} \; \forall i \in c_j, \forall j \in \{1,\cdots,|\mathcal{C}|\}. \tag{49}$$

It can be seen that the LP Q′ is similar to the LPs P'_j and only differs in that the upper bounds of the instance weights $D_{t+1}(i)$ are scaled by an amount d_j (for $\mathbf{x}_i \in c_j$) which is inversely proportional to the average hinge loss obtained for the corresponding class in the first stage of LPs. In other words, greater regularization is induced (by enforcing a lower upper bound for instance weights) for the class having a greater proportion of noisy and outlier instances, while also maintaining higher weightage for the non-outlier minority class instances to compensate for class imbalance.

Based on these two dual formulations, we now present the complete Dual-LexiBoost method as Algorithm 3.

```
Algorithm 3: Dual-LexiBoost
Input: Dataset $X$ of labelled instances $\mathbf{x}_i$.
Output: Final ensemble classifier
        $H(\mathbf{x}_i) = \text{sign}(\sum_{t=1}^{T}\alpha_t^* f_t(\mathbf{x}_i))$.
Initialize $D_1(i) = 1/n_j \; \forall i \in c_j, \forall j \in \{1,\cdots,|\mathcal{C}|\}$.
for all $t = 1$ to $T$ do
    Train weak learner $f_t$ using distribution $D_t$.
    Calculate training error $\epsilon_t$ using distribution $D_t$.
    if $\epsilon_t > \frac{1}{|\mathcal{C}|}$ then
        break
    end if
    for all $j \in \{1,\cdots,|\mathcal{C}|\}$ do
        Solve LP $\text{P}'_j$ to obtain $D_{t+1}(i) \; \forall i \in c_j$.
    end for
end for
for all $j \in \mathcal{C}$ do
    Solve LP $P_j$ to obtain $\boldsymbol{\lambda}_j^*$.
end for
Initialize $D_1(i) = 1/n_j \; \forall i \in c_j, \forall j \in \{1,\cdots,|\mathcal{C}|\}$.
for all $t = 1$ to $T$ do
    Train weak learner $f_t$ using distribution $D_t$.
    Calculate training error $\epsilon_t$ using distribution $D_t$.
    if $\epsilon_t > \frac{1}{|\mathcal{C}|}$ then
        break
    end if
    Solve LP Q′ to obtain $D_{t+1}(i) \; \forall i \in \{1,\cdots,n\}$.
end for
Solve LP Q to obtain $\boldsymbol{\alpha}^*$.
```

3.4 Generalization to multi-class tasks

It is easy to see that the proposed approach is readily applicable to polychotomous or multi-class classification tasks. However, the definition of margin must be altered for the multi-class setting in the following way:

Definition 4. *Let $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ be a given multi-class training dataset with label vectors $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{C}|}$ such that*

$$y_{i,j} = \begin{cases} 1 \text{ if } \mathbf{x}_i \text{ belongs to the } j\text{-th class}, \\ -1 \text{ otherwise}. \end{cases}$$

Additionally, let $\mathbf{f}_t(\mathbf{x}_i)$ denote the prediction vector for the point $\mathbf{x}_i$ by the classifier f_t. Then the margin $\rho(\mathbf{x}_i)$ can be redefined as

$$\rho(\mathbf{x}_i) = \sum_{t=1}^{T}\alpha_t \mathbf{y}_i{}^T \mathbf{f}_t(\mathbf{x}_i).$$

Thereafter, LexiBoost as well as Dual-LexiBoost can be directly applied to multi-class problems by modifying the LPs P_j, Q, P'_j, and Q′ accordingly.

4 Experiments

In this section, we report the results of experiments conducted on two-class artificial datasets of varying specifications, two-class as well as multi-class real-world datasets, multi-class hyperspectral image classification, and multi-class classification of a class imbalanced subset of the ImageNet dataset. We compare our results with AdaBoost (the AdaBoost.M2 variant [24] being used for multi-class datasets) which serves as a baseline, and with a shifted and calibrated [25] AdaBoost variant known as AdaMEC-Calib which has recently been found to be quite effective for imbalanced datasets [6]. We also compare our results with those of the primal solutions (i.e. using classifiers already created by AdaBoost) as well as dual solutions to LPAdaBoost (because of its inherent ability to tackle imbalance in noise-free situations) and LPUBoost. The C4.5 decision tree [26] and the k-Nearest Neighbor (kNN) classifier are used as base learners for the experiments on artificial and real-world datasets while only kNN is used as the base learner for hyperspectral image and ImageNet classification, due to the relatively large scale (in terms of the number of points or features) of the latter datasets. The parameters for C4.5 are chosen as per [26] while the parameter k for kNN is varied in the range $\{3, 5, 10\}$ for all sets of experiments except for hyperspectral image classification, where $k = 10$ is used because of the relatively large size of the datasets. The cost of false negative for AdaMEC-Calib is varied as elements in $\{2, 5, 10\}$ [15] and 50% of the training data is used for calibrating AdaMEC-Calib [6], The value β for LPUBoost is varied as elements in $\{2, 4, 8\}$ and D_{LB} is varied in the set $\{25, 50, 100\}$. $D = \frac{1}{\nu}$ is used for both LPAdaBoost and LPUBoost, with ν being varied as elements in $\{0.1, 0.2\}$ [18]. The performance is reported in terms of G-Mean [27], AUC [28], and Avg-AUC [29]. Only the best results obtained by each algorithm (chosen in terms of average AUC or Avg-AUC values over 5 runs of 5-fold cross-validation) are reported.

4.1 Artificial Datasets

We create 27 two-class artificial datasets by sampling points from two distinct 5-dimensional standard normal distributions by varying the Imbalance Ratio (IR) in $\{5, 10, 25\}$ and the total size of the datasets in $\{500, 1000, 2500\}$. The overlap between the classes is also varied by keeping the centre for the minority class fixed at $[0, 0, 0, 0, 0]^T$ while the centre for the majority class is varied between $[3, 3, 3, 3, 3]^T$, $[1.7, 1.7, 1.7, 1.7, 1.7]^T$, and $[1.5, 1.5, 1.5, 1.5, 1.5]^T$. 27 noisy analogues of these are also created by replacing 10% of the instances of each class with instances from the opposite class. The G-Mean values obtained for these 54 datasets are summarized from different perspectives for C4.5 as well as kNN as base learner in Figures 4-7 and 8-11, respectively.

We first look at the performance of the different algorithms when C4.5 is used as the base classifier. A look at Figures 4-7 shows that LexiBoost performs comparably to the other primal methods viz. LPAdaBoost and LPUBoost for the less challenging datasets having low IR, large size and low overlap. For the more challenging cases, LexiBoost outperforms the other primal methods as well as AdaBoost and AdaMEC-Calib. On the other hand, Dual-LexiBoost seems to perform better than the other dual

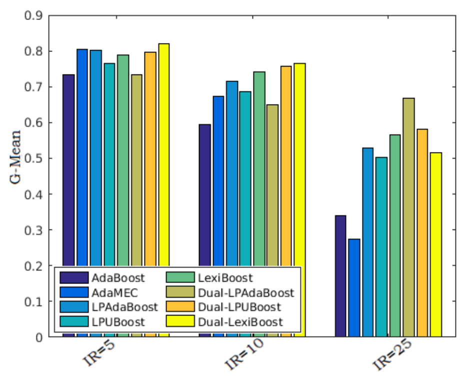


Fig. 4. Average G-Mean values for varying IRs for C4.5.

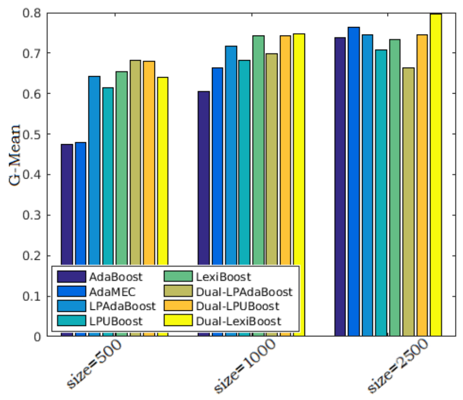


Fig. 5. Average G-Mean values for varying dataset sizes for C4.5.

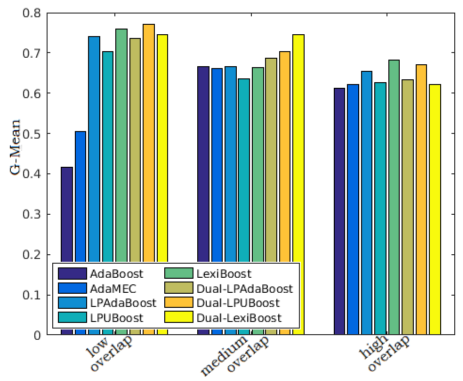


Fig. 6. Average G-Mean values for varying overlap for C4.5.

formulations viz. Dual-LPAdaBoost and Dual-LPUBoost for the less and moderately challenging datasets, with the performance faltering slightly for the more challenging datasets having high IR, small size and high overlap. However, it consistently maintains better performance than both AdaBoost and AdaMEC-Calib. The primal methods seem to perform slightly better than the dual variants for the non-noisy datasets, with LPAdaBoost achieving the best performance as expected. For the noisy datasets, AdaMEC-Calib and Dual-LPUBoost, equipped with exhaustive cost set tuning, perform the best. However, among the other methods,

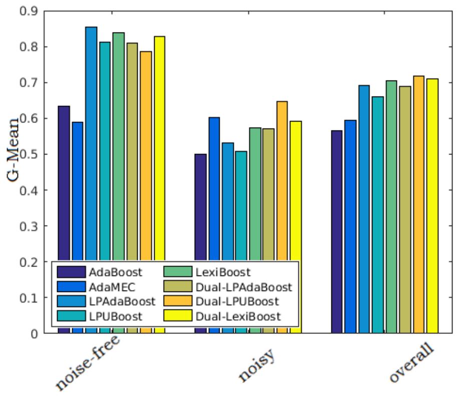


Fig. 7. Average G-Mean values for low noise and high noise cases for C4.5.

Dual-LexiBoost and LexiBoost perform the best. This points towards the effectiveness of the proposed regularization scheme. Overall, both LexiBoost and Dual-LexiBoost perform very well, being comparable to the cost set tuning aided Dual-LPUBoost, which achieves the best overall performance.

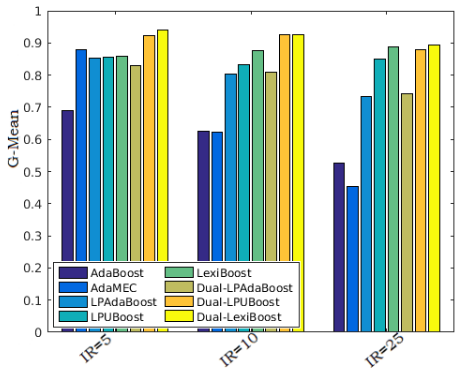


Fig. 8. Average G-Mean values for varying IRs for kNN.

A perusal of Figures 8-11, which show the performance with kNN being used as the base classifier, presents a slightly different picture. Unlike the performance with C4.5, when kNN is used as the underlying classifier, LexiBoost consistently performs better than the other primal techniques as well as the AdaBoost baseline and AdaMEC-Calib. Similarly, Dual-LexiBoost consistently performs best among all the algorithms. Moreover, Dual-LexiBoost even outperforms the exhaustive cost tuning methods, namely AdaMEC-Calib, LPUBoost and Dual-LPUBoost on noisy datasets. It is interesting to note that the dual variants seem to generally perform better than their primal counterparts.

Considering the performance for both C4.5 and kNN together makes it clear that the proposed LexiBoost method as well as its dual counterpart (both of which do not resort to cost set tuning) are very effective in handling class imbalanced classification problems due to their immunity to high degree of imbalance, small size of the dataset, overlap between the classes, as well as noise and outliers.

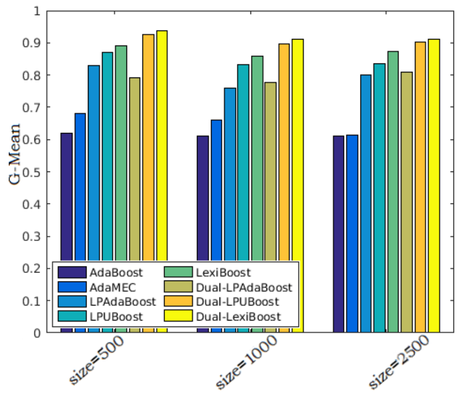


Fig. 9. Average G-Mean values for varying dataset sizes for kNN.

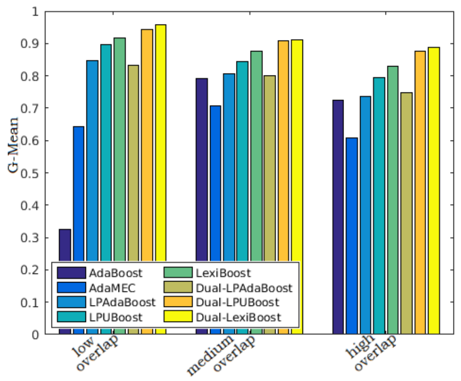


Fig. 10. Average G-Mean values for varying overlap for kNN.

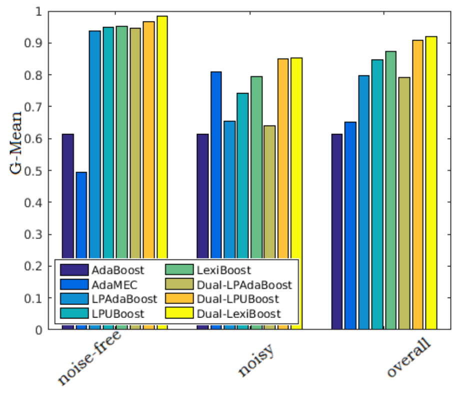


Fig. 11. Average G-Mean values for low noise and high noise cases.

4.2 Real-world Datasets

For the experiments on real-world imbalanced datasets, we use 20 two-class and 10 multi-class datasets with varying degrees of imbalance from the KEEL repository [30]. All the methods are compared on the two-class datasets while only AdaBoost (the AdaBoost.M2 variant), LPAdaBoost, LexiBoost, Dual-LPAdaBoost and Dual-LexiBoost are compared on the multi-class datasets (as the other methods are not directly adaptable to multi-class problems).

TABLE 1
AUC and G-Mean (GM) values with Average Ranking for two-class datasets with C4.5 as base learner.

Dataset	IR	Index	AdaBoost	AdaMEC-Calib	LPAdaBoost	LPUBoost	LexiBoost	Dual-LPAdaBoost	Dual-LPUBoost	Dual-LexiBoost
ecoli0vs1	1.85	AUC	0.8299	0.9768	0.9802	0.9698	**0.9837**	0.8066	0.9698	**0.9837**
		GM	0.6632	0.9763	0.9799	0.9693	**0.9834**	0.6394	0.9693	**0.9834**
german	2.34	AUC	0.5489	0.5418	0.5501	0.5510	**0.5543**	0.5194	0.5357	0.5523
		GM	0.2529	**0.4295**	0.3262	0.3278	0.3915	0.1958	0.3007	0.3532
new-thyroid2	5.14	AUC	0.7932	0.9028	0.9293	0.9293	0.9287	0.6361	**0.9528**	**0.9528**
		GM	0.6262	0.8990	0.9280	0.9280	0.9266	0.3017	**0.9515**	**0.9515**
yeast3	8.10	AUC	0.8493	0.8870	0.7982	0.8147	0.8577	0.6924	**0.9363**	0.9261
		GM	0.8405	0.8828	0.7787	0.7988	0.8509	0.6510	**0.9360**	0.9255
ecoli3	8.60	AUC	0.7149	0.8346	0.7807	0.7807	**0.8363**	0.5869	0.8045	0.7946
		GM	0.6633	0.8266	0.7621	0.7621	**0.8292**	0.4924	0.7939	0.7904
glass2	11.59	AUC	0.5152	0.5000	0.5000	0.5177	0.5993	0.5379	0.6216	**0.6375**
		GM	0.1361	0.1271	0.1274	0.2878	0.4383	0.3247	0.6000	**0.6313**
soybean12	14.52	AUC	0.5000	0.6667	0.9667	0.9801	0.9778	0.8687	0.9969	**0.9992**
		GM	0.0000	0.3333	0.9648	0.9779	0.9770	0.8642	0.9969	**0.9992**
sick2	15.33	AUC	0.9234	0.7472	0.9134	0.9176	**0.9437**	0.6345	0.7985	0.7639
		GM	0.9209	0.6943	0.9102	0.9147	**0.9430**	0.2994	0.7165	0.6206
glass4	15.46	AUC	0.5950	0.5000	0.7884	0.7934	**0.8476**	0.6735	0.8326	0.8371
		GM	0.2563	0.1468	0.7655	0.7693	**0.8321**	0.4966	0.8208	0.8257
shuttle-C2vsC4	20.50	AUC	0.5000	0.5833	**0.9167**	**0.9167**	**0.9167**	**0.9167**	**0.9167**	**0.9167**
		GM	0.0000	0.2357	**0.9024**	**0.9024**	**0.9024**	**0.9024**	**0.9024**	**0.9024**
glass5	22.78	AUC	0.5556	0.5000	0.7075	0.9126	0.9126	0.6111	0.8015	**0.9273**
		GM	0.1925	0.0000	0.5107	0.9062	0.9062	0.2722	0.6340	**0.9213**
yeast4	28.10	AUC	0.5525	0.5154	0.5767	0.6240	0.6845	0.4635	**0.7283**	0.6985
		GM	0.3213	0.1133	0.3972	0.5020	0.6560	0.2260	**0.7064**	0.6797
yeast5	32.73	AUC	0.7133	0.5289	0.7021	0.8558	0.9236	0.5000	0.9358	**0.9507**
		GM	0.6291	0.4141	0.6135	0.8413	0.9154	0.0000	0.9304	**0.9495**
wineQuality-red8vs67	46.50	AUC	0.5272	0.5000	0.5532	0.6010	**0.6153**	0.5140	0.5699	0.5451
		GM	0.1361	0.0000	0.2712	**0.4590**	0.4478	0.1304	0.4437	0.3259
poker8vs6	85.88	AUC	0.5000	0.5000	0.5179	0.5179	0.5499	0.5082	0.5555	**0.5643**
		GM	0.0000	0.0000	0.1425	0.1425	0.4990	0.1413	0.5008	**0.5243**
Average Rank		AUC	6.23	6.33	4.83	3.93	2.50	6.83	3.03	**2.30**
		GM	6.37	6.03	4.93	3.93	2.63	6.77	2.97	**2.37**

Best values shown in **boldface**.

TABLE 2
AUC and G-Mean (GM) values with Average Ranking for two-class datasets with kNN as base learner.

Dataset	IR	Index	AdaBoost	AdaMEC-Calib	LPAdaBoost	LPUBoost	LexiBoost	Dual-LPAdaBoost	Dual-LPUBoost	Dual-LexiBoost
ecoli0vs1	1.85	AUC	0.8299	0.9666	0.9766	0.9837	0.9837	**0.9872**	**0.9872**	**0.9872**
		GM	0.6632	0.9664	0.9763	0.9834	0.9834	**0.9869**	**0.9869**	**0.9869**
german	2.34	AUC	0.5806	0.5802	0.5764	0.5810	0.6125	0.5992	**0.6192**	0.6164
		GM	0.5036	0.5152	0.5294	0.5957	0.6109	0.5764	**0.6140**	0.6133
new-thyroid2	5.14	AUC	0.9460	0.9417	0.9778	0.9778	0.9861	0.9778	0.9778	**0.9889**
		GM	0.9454	0.9391	0.9775	0.9775	0.9860	0.9775	0.9775	**0.9888**
yeast3	8.10	AUC	0.8302	0.8316	0.8308	0.8308	0.8367	0.8017	0.8951	**0.9132**
		GM	0.8167	0.8214	0.8200	0.8200	0.8340	0.7875	0.8933	**0.9131**
ecoli3	8.60	AUC	0.7998	0.6245	0.7998	0.7998	0.7998	0.8083	**0.8855**	0.8788
		GM	0.7756	0.3818	0.7756	0.7756	0.7756	0.7898	**0.8840**	0.8776
glass2	11.59	AUC	0.6064	0.7128	0.6516	0.7118	0.7769	0.6516	0.7567	**0.7899**
		GM	0.5009	0.7083	0.5811	0.6535	0.7734	0.6160	0.7474	**0.7828**
soybean12	14.52	AUC	0.6667	0.6444	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
		GM	0.3333	0.3103	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
sick2	15.33	AUC	0.6404	0.7209	0.6447	0.6508	**0.7456**	0.5952	0.7047	0.7158
		GM	0.5666	0.7203	0.5740	0.6131	**0.7449**	0.4955	0.6941	0.7074
glass4	15.46	AUC	0.7601	0.8986	0.7650	0.8526	0.8918	0.8292	0.9359	**0.9409**
		GM	0.7233	0.8937	0.7033	0.8375	0.8886	0.7887	0.9337	**0.9377**
shuttle-C2vsC4	20.50	AUC	0.5000	0.4167	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
		GM	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
glass5	22.78	AUC	0.8211	0.7657	0.9250	**0.9634**	**0.9634**	0.8645	0.9347	0.9585
		GM	0.7882	0.5937	0.9191	**0.9626**	**0.9626**	0.8564	0.9295	0.9575
yeast4	28.10	AUC	0.6692	**0.8174**	0.6128	0.7743	0.7999	0.6601	0.7934	0.7922
		GM	0.5934	**0.8153**	0.4734	0.7583	0.7963	0.5827	0.7817	0.7793
yeast5	32.73	AUC	0.8348	0.7111	0.8281	0.8334	0.8557	0.8006	0.9599	**0.9736**
		GM	0.8189	0.5335	0.8002	0.8172	0.8432	0.7726	0.9591	**0.9734**
wineQuality-red8vs67	46.50	AUC	0.5164	0.5000	0.5490	0.5648	0.5612	0.5693	**0.5717**	0.5672
		GM	0.1339	0.0000	0.3345	0.3994	0.4641	0.4455	0.4465	**0.4688**
poker8vs6	85.88	AUC	0.5549	0.5000	0.6074	0.6074	0.6088	0.**6228**	0.6084	0.6054
		GM	0.2839	0.0000	0.4733	**0.4739**	**0.4739**	0.4208	0.4737	0.4722
Average Rank		AUC	6.77	6.13	5.70	4.47	3.03	4.80	2.63	**2.47**
		GM	6.87	6.10	5.63	4.27	2.80	5.30	2.77	**2.27**

Best values shown in **boldface**.

4.2.1 Two-class Classification

The performance over the 20 two-class real-world imbalanced datasets are summarized in Tables 1 and 2 for C4.5 and kNN as the base learner, respectively. The results are reported in terms of G-Mean and AUC values along with the average ranks. The signed rank test hypotheses [31] for Tables 1 and 2 are reported in Tables 3 and 4, respectively. It is seen that Dual-LexiBoost achieves the best rank, followed respectively by LexiBoost and Dual-LPUBoost for the C4.5 based experiments. This is in contrast to the results observed for the artificial datasets in Section 4.1 indicating that despite the slightly lower performance on the more difficult artificial datasets (when used in conjunction with C4.5), Dual-LexiBoost is better able to handle imbalance compared to both LexiBoost as well as Dual-LPUBoost in practice. The performances of Dual-LexiBoost, LexiBoost and Dual-LPUBoost are found to be statistically equivalent in terms of both G-Mean and AUC for C4.5, while Dual-LPUBoost is significantly worse than Dual-LexiBoost in terms of G-Mean for the kNN based experiments. All the other contenders are found to perform significantly worse than both Dual-LexiBoost and LexiBoost for both choices of base learners. The lower average rankings achieved by Dual-LexiBoost indicate that the dual formulation for adapting instance weights is indeed useful for achieving better performance.

TABLE 3
Results of the Wilcoxon Signed Rank Test for Table 1

Control	LexiBoost		Dual-LexiBoost	
Index	AUC	G-Means	AUC	G-Means
AdaBoost	H_1	H_1	H_1	H_1
AdaMEC-Calib	H_1	H_1	H_1	H_1
LPAdaBoost	H_1	H_1	H_1	H_1
LPUBoost	H_1	H_1	H_0	H_0
LexiBoost	-	-	H_0	H_0
Dual-LPAdaBoost	H_1	H_1	H_1	H_1
Dual-LPUBoost	H_0	H_0	H_0	H_0
Dual-LexiBoost	H_0	H_0	-	-

H_0 : Control performs similar to contender.
H_1 : Significant difference between control and contender.

TABLE 4
Results of the Wilcoxon Signed Rank Test for Table 2

Control	LexiBoost		Dual-LexiBoost	
Index	AUC	G-Means	AUC	G-Means
AdaBoost	H_1	H_1	H_1	H_1
AdaMEC-Calib	H_1	H_1	H_1	H_1
LPAdaBoost	H_1	H_1	H_1	H_1
LPUBoost	H_1	H_1	H_1	H_1
LexiBoost	-	-	H_0	H_0
Dual-LPAdaBoost	H_1	H_1	H_1	H_1
Dual-LPUBoost	H_0	H_0	H_0	H_1
Dual-LexiBoost	H_0	H_0	-	-

H_0 : Control performs similar to contender.
H_1 : Significant difference between control and contender.

4.2.2 Multi-class Classification

The performance of AdaBoost.M2, LPAdaBoost, LexiBoost, Dual-LPAdaBoost and Dual-LexiBoost (on the multi-class real-world datasets having varying number of classes and degree of imbalance) is summarized for C4.5 and kNN respectively in Tables 5 and 6. The results are reported in terms of the Avg-AUC and G-Mean values and the average ranks alongside the IRs (measured as the ratio of the number of representatives from the largest and the smallest classes). The corresponding hypotheses for the signed rank test [31] are reported in Tables 7 and 8, respectively. For the C4.5 based results in Table 5, both LexiBoost and Dual-LexiBoost attain the best average rank in terms of Avg-AUC. However, in terms of G-Mean, Dual-LexiBoost outperforms LexiBoost indicating that the former is able to strike a better balance between all the classes in the datasets. Indeed, the performance of LexiBoost in terms of G-Mean is also statistically similar to that of the AdaBoost.M2 baseline. However, Dual-LexiBoost is seen to perform significantly better than all methods other than LexiBoost in terms of both Avg-AUC and G-Mean. For the kNN based experiments, Dual-LexiBoost achieves the best rank followed by LexiBoost. While the performance of Dual-LexiBoost is significantly better than that of all other methods in terms of both indexes, the performance of LexiBoost is statistically similar to that of Dual-LPAdaBoost in terms of G-Mean, indicating that the dual method of instance weight adaptation is useful for generating better balance between the classes for multi-class datasets as well. It is also interesting to note that unlike LPAdaBoost and Dual-LPAdaBoost which perform worse than the AdaBoost.M2 baseline for the datasets having high number of classes, the performance of both LexiBoost and Dual-LexiBoost does not seem to be adversely affected by increase in the number of classes.

4.3 Hyperspectral Image Classification

Hyperspectral image classification has been listed, in a recent survey by Krawczyk [32], as one of the key practical application areas where multi-class imbalance naturally arises. Therefore, in this section, we test the effectiveness of the proposed techniques for this application. For the experiments on hyperspectral image classification, we use the Samson and Jasper ridge images from [33], the Kennedy Space Center (KSC) image from [34], and the Salinas A and Indian pines scenes from [35]. For the images which contain unlabeled pixels, the training and testing is only undertaken on the labeled pixels as per [36]. We only use kNN (with $k = 10$) as the base learner for these experiments because of the large size of these datasets. The results are listed in Table 9 in terms of Avg-AUC and G-Mean values. Since there are only a small number of images, we refrain from using the signed rank test but instead used the rank sum test [31], [37] for ascertaining the statistical significance of the results. It is seen that the proposed Dual-LexiBoost method achieves the best rank, followed by LexiBoost. Though the improvement achieved by LexiBoost is not deemed statistically significant while that of Dual-LexiBoost is only deemed statistically significant for the Indian pines image, the results obtained by LexiBoost and Dual-LexiBoost are visibly better than those of the contending methods as illustrated in Figure 12 for the KSC image.

4.4 ImageNet Classification

One of the more challenging applications of pattern classification is the classification of natural images. Uncurated natural image datasets are inherently class imbalanced. Moreover, the class distributions for such datasets are generally complex, making it learning algorithms more sensitive to issues such as cost set tuning and outlier regularization. Therefore, in this section, we undertake the classification of imbalanced subsets of the popular ImageNet dataset [38]. We prepare 3 datasets, namely ImageNet8, ImageNet9 and ImageNet12 for this purpose. The datasets are

TABLE 5
Avg-AUC and G-Mean (GM) values with Average Rankings for multi-class datasets with C4.5 as base learner.

Dataset	$\|\mathcal{C}\|$	IR	Index	AdaBoost.M2	LPAdaBoost	LexiBoost	Dual-LPAdaBoost	Dual-LexiBoost
wine	3	1.48	Avg-AUC	0.5000	0.9250	**0.9370**	**0.9370**	**0.9370**
			GM	0.0000	0.8927	**0.9091**	**0.9091**	**0.9091**
hayes-roth	3	1.70	Avg-AUC	0.8275	0.7711	**0.8289**	0.5049	0.8111
			GM	0.7573	0.6665	**0.7629**	0.0000	0.7318
contraceptive	3	1.89	Avg-AUC	0.5440	0.5140	0.5688	0.4693	**0.5704**
			GM	0.1673	0.0000	0.2226	0.2445	**0.4135**
new-thyroid	3	5.00	Avg-AUC	0.8885	0.6462	0.8937	0.6946	**0.9062**
			GM	0.8469	0.3242	0.8531	0.2965	**0.8716**
thyroid	3	39.18	Avg-AUC	0.6316	0.6776	0.9210	0.7196	**0.9352**
			GM	0.2824	0.3308	0.8837	0.4338	**0.9091**
lymphography	4	27	Avg-AUC	0.7864	0.6667	0.7690	0.7378	**0.7872**
			GM	0.2925	0.0000	0.2832	0.2643	**0.2975**
pageblocks	5	164	Avg-AUC	0.8424	0.7137	0.8526	0.6831	**0.8644**
			GM	0.4779	0.0000	0.4779	0.2260	**0.5479**
shuttle	5	568.67	Avg-AUC	0.6667	0.7107	**0.9574**	0.8740	0.9254
			GM	0.3332	0.1904	**0.6657**	0.3326	0.6652
ecoli	8	71.5	Avg-AUC	0.6809	0.5832	**0.7102**	0.6002	0.6855
			GM	0.0000	0.0000	0.0000	0.0000	0.0000
yeast	10	92.60	Avg-AUC	0.6472	0.5774	**0.6610**	0.5737	0.6360
			GM	0.0000	0.0000	0.0000	0.0000	0.0000
Average Rank			Avg-AUC	3.30	4.30	**1.70**	4.00	**1.70**
			GM	3.25	4.20	2.25	3.50	**1.80**

Best values shown in **boldface**.

TABLE 6
Avg-AUC and G-Mean (GM) values with Average Rankings for multi-class datasets with kNN as base learner.

Dataset	$\|\mathcal{C}\|$	IR	Index	AdaBoost.M2	LPAdaBoost	LexiBoost	Dual-LPAdaBoost	Dual-LexiBoost
wine	3	1.48	Avg-AUC	**0.7836**	0.7572	0.7818	0.7815	0.7818
			GM	**0.6996**	0.6690	0.6959	0.6959	0.6959
hayes-roth	3	1.70	Avg-AUC	0.7167	0.7309	0.7441	0.7328	**0.7598**
			GM	0.6109	0.6055	**0.6868**	0.6236	0.6767
contraceptive	3	1.89	Avg-AUC	0.6176	0.6014	0.6176	0.5778	**0.6315**
			GM	0.4715	0.4522	0.4715	0.4270	**0.5037**
new-thyroid	3	5.00	Avg-AUC	0.8953	0.9425	0.9425	0.9425	**0.9597**
			GM	0.8552	0.9211	0.9211	0.9211	**0.9452**
thyroid	3	39.18	Avg-AUC	0.5822	0.5859	0.6090	0.5944	**0.6389**
			GM	0.1926	0.1247	0.3147	0.2060	**0.4406**
lymphography	4	27	Avg-AUC	0.8034	0.7957	0.8608	0.7944	**0.8930**
			GM	0.2878	0.2841	0.6033	0.2755	**0.8229**
pageblocks	5	164	Avg-AUC	0.7540	0.8010	0.8083	0.7719	**0.8719**
			GM	0.2495	0.2495	0.2495	0.2495	**0.5272**
shuttle	5	568.67	Avg-AUC	0.8068	0.9308	0.9524	0.9320	**0.9529**
			GM	0.6152	0.6142	**0.6623**	0.6622	0.6617
ecoli	8	71.5	Avg-AUC	0.7953	0.7855	0.8353	0.7907	**0.8647**
			GM	0.0000	0.0000	0.2878	0.2656	**0.5622**
yeast	10	92.60	Avg-AUC	0.7261	0.7099	0.7303	0.7109	**0.7524**
			GM	0.0000	0.0000	0.0000	0.1217	**0.5081**
Average Rank			Avg-AUC	3.75	4.10	2.20	3.80	**1.15**
			GM	3.55	4.30	2.40	3.25	**1.50**

Best values shown in **boldface**.

TABLE 7
Results of the Wilcoxon Signed Rank Test for 5

Control	LexiBoost		Dual-LexiBoost	
Index	Avg-AUC	G-Means	Avg-AUC	G-Means
AdaBoost.M2	H_1	H_0	H_1	H_1
LPAdaBoost	H_1	H_1	H_1	H_1
LexiBoost	-	-	H_0	H_0
Dual-LPAdaBoost	H_1	H_1	H_1	H_1
Dual-LexiBoost	H_0	H_0	-	-

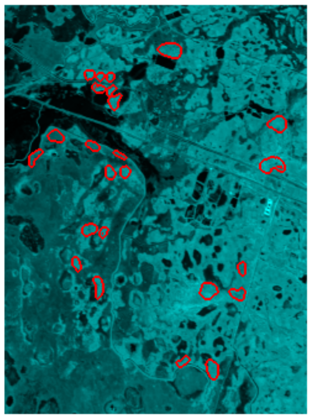
H_0 : Control performs similar to contender.
H_1 : Significant difference between control and contender.

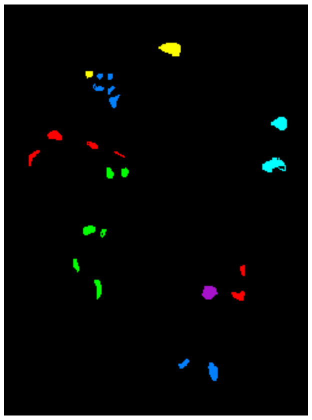
TABLE 8
Results of the Wilcoxon Signed Rank Test for 6

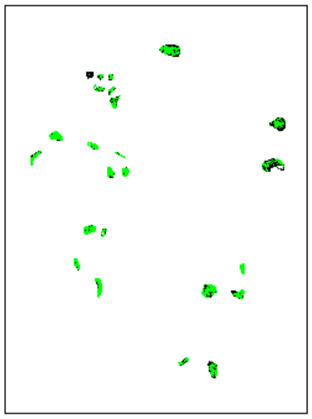
Control	LexiBoost		Dual-LexiBoost	
Index	Avg-AUC	G-Means	Avg-AUC	G-Means
AdaBoost.M2	H_1	H_1	H_1	H_1
LPAdaBoost	H_1	H_1	H_1	H_1
LexiBoost	-	-	H_1	H_1
Dual-LPAdaBoost	H_1	H_0	H_1	H_1
Dual-LexiBoost	H_1	H_1	-	-

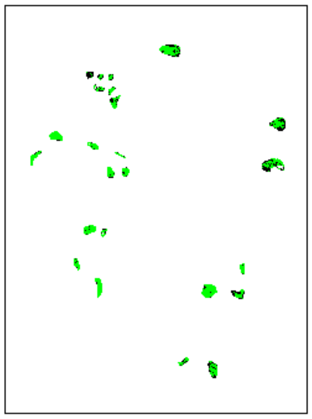
H_0 : Control performs similar to contender.
H_1 : Significant difference between control and contender.

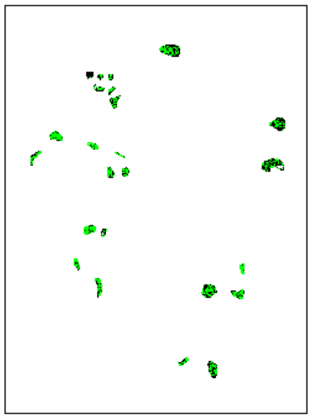
prepared by randomly choosing images corresponding to the 8 principal subtrees of the ImageNet dataset as well as the miscellaneous subtrees Foods, Collections, Documents, and Microorganisms, from the ImageNet 2011 Fall Release. The number of images collected from each of the subtrees corresponds to about 2% of the number of synsets contained within the subtree in question, with

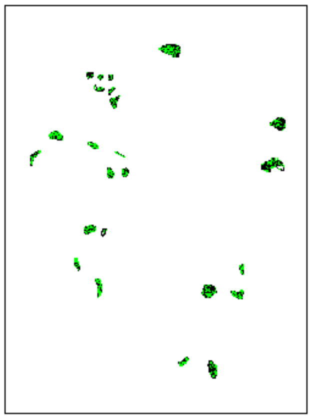

(a) Part of KSC image showing minority classes.


(b) Ground truth.


(c) LexiBoost.


(d) Dual-LexiBoost.


(e) AdaBoost.M2.


(f) LPAdaBoost.

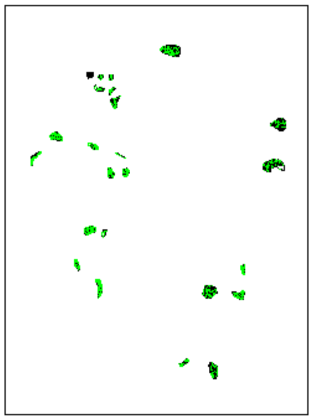

(g) Dual-LPAdaBoost.

Fig. 12. Comparison of the contending methods showing better performance of the proposal on minority classes. Misclassified points are shown in black and correctly-classified points are shown in green.

TABLE 9
Avg-AUC and G-Mean (GM) values with Average Rankings for hyperspectral images.

Dataset	$\|\mathcal{C}\|$	IR	Index	AdaBoost.M2	LPAdaBoost	LexiBoost	Dual-LPAdaBoost	Dual-LexiBoost
Samson	3	1.29	Avg-AUC	$0.9881^{\sim\approx}$	$0.9811^{\sim\approx}$	$0.9881^{\approx}$	$0.9865^{\sim\approx}$	$\mathbf{0.9892}^{\sim}$
			GM	$0.9842^{\sim\approx}$	$0.9746^{\sim\approx}$	$0.9842^{\approx}$	$0.9820^{\sim\approx}$	$\mathbf{0.9855}^{\sim}$
Jasper ridge	4	4.64	Avg-AUC	$0.9668^{\sim\approx}$	$0.9577^{\sim\approx}$	$0.9691^{\approx}$	$0.9671^{\sim\approx}$	$\mathbf{0.9777}^{\sim}$
			GM	$0.9490^{\sim\approx}$	$0.9358^{\sim\ddagger}$	$0.9507^{\approx}$	$0.9498^{\sim\approx}$	$\mathbf{0.9665}^{\sim}$
Salinas A	6	3.90	Avg-AUC	$0.9931^{\sim\approx}$	$0.9927^{\sim\approx}$	$0.9941^{\approx}$	$0.9935^{\sim\approx}$	$\mathbf{0.9956}^{\sim}$
			GM	$0.9884^{\sim\approx}$	$0.9878^{\sim\approx}$	$0.9905^{\approx}$	$0.9891^{\sim\approx}$	$\mathbf{0.9927}^{\sim}$
KSC	13	8.83	Avg-AUC	$0.8788^{\sim\approx}$	$0.8494^{\dagger\ddagger}$	$0.8928^{\approx}$	$0.8766^{\sim\approx}$	$\mathbf{0.9015}^{\sim}$
			GM	$0.7511^{\sim\approx}$	$0.6920^{\dagger\ddagger}$	$0.7780^{\approx}$	$0.7476^{\sim\approx}$	$\mathbf{0.8030}^{\sim}$
Indian pines	16	122.75	Avg-AUC	$0.8401^{\sim\ddagger}$	$0.8358^{\sim\ddagger}$	$0.8395^{\ddagger}$	$0.8277^{\sim\ddagger}$	$\mathbf{0.8641}^{\dagger}$
			GM	$0.6667^{\sim\ddagger}$	$0.6577^{\sim\ddagger}$	$0.6604^{\ddagger}$	$0.6066^{\dagger\ddagger}$	$\mathbf{0.7367}^{\dagger}$
Average Rank			Avg-AUC	3.10	4.80	2.30	3.80	**1.0000**
			GM	3.10	4.80	2.30	3.80	**1.0000**

$\sim$, $\approx$: Statistically similar to LexiBoost and Dual-LexiBoost, respectively.
$\dagger$, $\ddagger$: Significantly different than LexiBoost and Dual-LexiBoost, respectively.
Best values shown in **boldface**.

the constraint that at least 20 images must be chosen from each subtree. The data sampling hierarchy thus obtained is illustrated in Figure 13. The dataset ImageNet8 is prepared by only combining the samples from the 8 principal subtrees and not including the miscellaneous images, giving rise to a dataset containing 1120 images. The ImageNet9 dataset adds to the complexity of the classification task by appending the 120 miscellaneous images as a single class, resulting in a dataset of size 1240. The complexity is increased further in the ImageNet12 dataset as the images belonging to the miscellaneous subtrees Foods, Collections, Documents and Microorganisms are classified into 4 different classes corresponding to these subtrees. In keeping with the state-of-the-art in feature representation of images, we derive a 2048-dimensional deep feature space representation of each image from the final global average pooling layer of the Inception-v3 deep neural network [39]. We only use kNN as the base learner for these experiments because of the large dimensionality of these datasets.

The results achieved by AdaBoost.M2, LPAdaBoost, LexiBoost, Dual-LPAdaBoost and Dual-LexiBoost are detailed in Table 10 in terms of Avg-AUC and G-Mean along with the average ranks and rank sum test results. Yet again, Dual-LexiBoost is observed to achieve the best rank followed by LexiBoost. However, the interesting thing to observe in this set of results is that as the complexity increases from ImageNet8 to ImageNet9, the performance of AdaBoost.M2, LPAdaBoost and Dual-LPAdaBoost falls drastically as evident from the G-Mean values. Yet the performance of LexiBoost is still better, showing that LexiBoost is indeed able to strike a balance between the classes by assigning high weightage to the component classifiers which favor the minority classes; something that AdaBoost.M2 and LPAdaBoost fail to do. The even better performance attained by Dual-LexiBoost on ImageNet9 demonstrates the efficacy of the dual formulation in generating proper instance weights. This becomes even more evident as one observes the results for ImageNet12. All the contenders other than Dual-LexiBoost fails in terms of G-Mean on this dataset, indicating that the component classifiers generated by both AdaBoost.M2 and Dual-LPAdaBoost have not been able to properly classify some of the (presumably) minority classes. However, even on this dataset, Dual-LexiBoost is able to achieve a decent performance owing to the efficacy of its instance weight selection scheme.

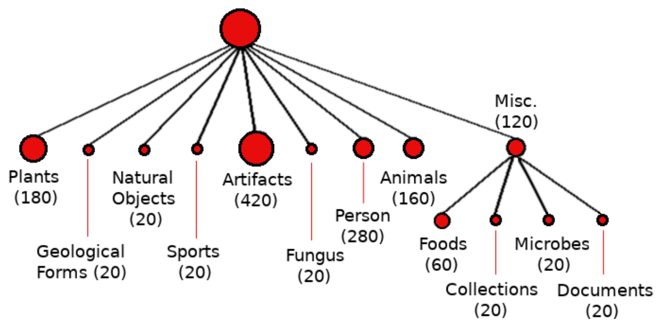


Fig. 13. ImageNet subset hierarchy.

5 Conclusions

Based on the understanding that the choice of component classifier weights for boosting can be thought of as a game of *Tug of War* between the classes in the margin space, we introduce the reader to a two-staged LLP framework for choosing component weights to handle class imbalance. The proposed framework is called LexiBoost. LexiBoost introduces a novel regularization scheme which offers some advantages over the traditional slack variable reliant scheme, due to the fact that the proposed scheme does not require to undertake expensive cost set tuning which has been the norm for imbalanced classification till date. Hence, the proposed framework solves the long-standing problem of cost set tuning for imbalanced classification which had hindered the direct extension of these methods to multi-class problems. This makes LexiBoost directly applicable to both dichotomous as well as polychotomous tasks. We also derive the dual algorithm corresponding to the proposed method, called Dual-LexiBoost. Experiments conducted on artificial datasets, real-world imbalanced datasets and hyperspectral images suggest that the proposed methods exhibit greater immunity to class imbalance, overlap, size of the dataset, noise, as well as the number of classes in the dataset. Moreover, the Dual-LexiBoost is observed to be able to generate good component classifiers even when the primal methods fail due to the complexity of the datasets (as observed from the experiments on the challenging ImageNet dataset). In the near future, the authors plan to extend the proposed framework to single-class classification along the lines of [40].

Acknowledgement

We would like to thank Anubhav Agrawal, pre-final year student pursuing B.Tech. in Electronics and Electrical Engineering from the Indian Institute of Technology, Guwahati, India, for his extensive help with the computer implementations of the different techniques used in our experiments.

References

[1] R. E. Schapire, “The strength of weak learnability,” *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.
[2] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee *et al.*, “Boosting the margin: A new explanation for the effectiveness of voting methods,” *The annals of statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.
[3] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, “Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance,” *Neural networks*, vol. 21, no. 2, pp. 427–436, 2008.
[4] C. Phua, D. Alahakoon, and V. Lee, “Minority report in fraud detection: Classification of skewed data,” *Acm sigkdd explorations newsletter*, vol. 6, no. 1, pp. 50–59, 2004.
[5] C. Huang, Y. Li, C. C. Loy, and X. Tang, “Learning deep representation for imbalanced classification,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5375–5384.
[6] N. Nikolaou, N. Edakunni, M. Kull, P. Flach, and G. Brown, “Cost-sensitive boosting algorithms: Do we really need them?” *Machine Learning*, vol. 104, no. 2-3, pp. 359–384, 2016.
[7] Y. Freund and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” in *European conference on computational learning theory*. Springer, 1995, pp. 23–37.
[8] K. M. Ting, “A comparative study of cost-sensitive boosting algorithms,” in *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer, 2000.
[9] P. Viola and M. Jones, “Fast and robust classification using asymmetric Adaboost and a detector cascade,” *Advances in neural information processing systems*, vol. 2, pp. 1311–1318, 2002.
[10] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, “Cost-sensitive boosting for classification of imbalanced data,” *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.
[11] H. Masnadi-Shirazi and N. Vasconcelos, “Asymmetric boosting,” in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 609–619.
[12] ——, “Cost-sensitive boosting,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 33, no. 2, pp. 294–309, 2011.
[13] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, “A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.
[14] I. Landesa-Vázquez and J. L. Alba-Castro, “Double-base asymmetric Adaboost,” *Neurocomputing*, vol. 118, pp. 101–114, 2013.
[15] N. Nikolaou and G. Brown, “Calibrating Adaboost for asymmetric learning,” in *International Workshop on Multiple Classifier Systems*. Springer, 2015, pp. 112–124.
[16] A. J. Grove and D. Schuurmans, “Boosting in the limit: Maximizing the margin of learned ensembles,” in *AAAI/IAAI*, 1998, pp. 692–699.
[17] G. Rätsch, T. Onoda, and K.-R. Müller, “Soft margins for Adaboost,” *Machine learning*, vol. 42, no. 3, pp. 287–320, 2001.
[18] J. Leskovec and J. Shawe-Taylor, “Linear programming boosting for uneven datasets,” in *ICML*, 2003, pp. 456–463.
[19] F. Charro, J. G. Azorero, and J. D. Rossi, “A mixed problem for the infinity Laplacian via tug-of-war games,” *Calculus of Variations and Partial Differential Equations*, vol. 34, no. 3, pp. 307–320, 2009.

TABLE 10
Avg-AUC and G-Mean (GM) values with Average Rankings for ImageNet subset classification.

Dataset	$\|\mathcal{C}\|$	IR	Index	AdaBoost.M2	LPAdaBoost	LexiBoost	Dual-LPAdaBoost	Dual-LexiBoost
ImageNet8	8	21	Avg-AUC	$0.8292^{\sim\ddagger}$	$0.8053^{\dagger\ddagger}$	$0.8366^{\approx}$	$0.8296^{\sim\ddagger}$	$\mathbf{0.8534}^{\sim}$
			GM	$0.2424^{\sim\ddagger}$	$0.2094^{\dagger\ddagger}$	$0.2436^{\ddagger}$	$0.2237^{\dagger\ddagger}$	$\mathbf{0.4588}^{\dagger}$
ImageNet9	9	21	Avg-AUC	$0.7817^{\sim\ddagger}$	$0.7747^{\sim\ddagger}$	$0.7986^{\ddagger}$	$0.7660^{\dagger\ddagger}$	$\mathbf{0.8285}^{\dagger}$
			GM	$0.0000^{\dagger\ddagger}$	$0.0000^{\dagger\ddagger}$	$0.2180^{\ddagger}$	$0.0000^{\dagger\ddagger}$	$\mathbf{0.2481}^{\dagger}$
ImageNet12	12	21	Avg-AUC	$0.7676^{\sim\ddagger}$	$0.7477^{\dagger\ddagger}$	$0.7665^{\ddagger}$	$0.7558^{\sim\ddagger}$	$\mathbf{0.8323}^{\dagger}$
			GM	$0.0000^{\sim\ddagger}$	$0.0000^{\sim\ddagger}$	$0.0000^{\ddagger}$	$0.0000^{\sim\ddagger}$	$\mathbf{0.2349}^{\dagger}$
Average Rank			Avg-AUC	3.10	4.80	2.30	3.80	**1.0000**
			GM	3.10	4.80	2.30	3.80	**1.0000**

$\sim$, $\approx$: Statistically similar to LexiBoost and Dual-LexiBoost, respectively.
$\dagger$, $\ddagger$: Significantly different than LexiBoost and Dual-LexiBoost, respectively.
Best values shown in **boldface**.

[20] Y. Peres, O. Schramm, S. Sheffield, and D. B. Wilson, "Tug-of-war and the infinity Laplacian," in *Selected Works of Oded Schramm*. Springer, 2011, pp. 595–638.

[21] J. Sumpsi, F. Amador, and C. Romero, "On farmers' objectives: A multi-criteria approach," *European Journal of Operational Research*, vol. 96, no. 1, pp. 64–71, 1997.

[22] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[23] C. Shen and H. Li, "On the dual formulation of boosting algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2216–2231, 2010.

[24] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *icml*, vol. 96, 1996, pp. 148–156.

[25] J. Platt *et al.*, "Probabilistic outputs for Support Vector Machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[26] J. R. Quinlan, *C4. 5: Programs for machine learning*. Elsevier, 2014.

[27] M. Kubat, S. Matwin *et al.*, "Addressing the curse of imbalanced training sets: One-sided selection," in *ICML*, vol. 97. Nashville, USA, 1997, pp. 179–186.

[28] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *ICML-2003 workshop on learning from imbalanced data sets II*, vol. 2, 2003, pp. 2–1.

[29] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.

[30] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2010.

[31] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[32] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[33] F. Zhu, Y. Wang, B. Fan, S. Xiang, G. Meng, and C. Pan, "Spectral unmixing via data-guided sparsity," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5412–5427, 2014.

[34] T. V. Bandos, L. Bruzzone, and G. Camps-Valls, "Classification of hyperspectral images with regularized linear discriminant analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3, pp. 862–873, 2009.

[35] "Hyperspectral remote sensing scenes," http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes, accessed: 2017-08-24.

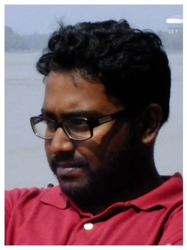
[36] T. Sun, L. Jiao, J. Feng, F. Liu, and X. Zhang, "Imbalanced hyperspectral image classification based on maximum margin," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 3, pp. 522–526, 2015.

[37] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[38] "ImageNet," http://www.image-net.org/, accessed: 2017-08-24.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[40] G. Rätsch, B. Schölkopf, S. Mika, and K.-R. Müller, "Constructing boosting algorithms from SVMs:an application to one-class classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1184–1199, 2002.



Shounak Datta received his B.Tech. degree in Electronics and Communication Engineering from the West Bengal University of Technology, Kolkata, India in 2011, and M.E. in Electronics and Telecommunication Engineering from the Jadavpur University, Kolkata, India in 2013. He is currently pursuing a Ph.D. in Computer Science from the Indian Statistical Institute, Kolkata, India. His research interests include imbalanced dataset classification, learning with missing features, multi-objective optimization in machine learning, etc.

Sayak Nag has recently completed his B.E. in Instrumentation and Electronics Engineering from the Jadavpur University, Kolkata, India. His research interests include ensembles classifiers, support vector machines, neural networks, multi-objective optimization, and machine learning in general.

Swagatam Das is currently serving as a faculty member at the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, India. He has published one research monograph, one edited volume, and more than 200 research articles in peer-reviewed journals and international conferences. He is the founding co-editor-in-chief of Swarm and Evolutionary Computation, an international journal from Elsevier. Dr. Das has 12,000+ Google Scholar citations and an H-index of 54 till date.