**International Journal of Intelligent Information Technologies**
Volume 18 • Issue 1

# A Hybrid Learning Framework for Imbalanced Classification

Eric P. Jiang, University of San Diego, USA*

## ABSTRACT

Class imbalance is a well-known and challenging algorithmic research topic among the machine learning community as traditional classifiers generally perform poorly on imbalanced problems, where data to be learned have skewed distributions between their classes. This paper presents a hybrid framework named PRUSBoost for learning imbalanced classification. It combines a selective data under-sampling procedure and a powerful boosting strategy to effectively enhance classification performance on imbalanced problems. Different from the simple random under sampling algorithm, this framework constructs the training data of the majority or negative class by using a newly developed partition based under sampling approach. Experiments on several datasets from different application domains that carry skewed class distributions have shown that the proposed framework provides a very competitive, consistent, and effective solution to imbalanced classification problems.

## KEYWORDS

Boosting, Data Sampling, Imbalanced Classification, Machine Learning, Novelty Detection

## INTRODUCTION

Class imbalance is a well-known and challenging problem among the machine learning community. It refers to the applications where data from different classes are noticeably unevenly distributed. For a binary classification problem, it means that samples from one class (that is usually called the majority or negative class) significantly outnumbers those from the other (that is named the positive or minority class). Traditional classification algorithms generally fail to work adequately with skewed class distribution problems. They are designed to generalize from sample data and produce the simplest hypothesis that best fits the data. This learning principle is represented as the inductive bias of some machine learning algorithms such as decision trees, which prefer small trees over large ones (Akabani et al., 2004). As a result of this, given an unbalanced data set, they often generate the hypothesis that classifies almost all its samples as negative.

Clearly, such a hypothesis can simply be useless in practice. For instance, assume we need to build a model with a data set that contains customer transaction records and among them, there are only a very tiny portion of transactions are confirmed fraudulent and the rest of transactions are

DOI: 10.4018/IJIIT.306967 *Corresponding Author

deemed as normal. To protect customers and their financial assets, we are most interested in detecting successfully as many fraudulent activities as possible. When facing this kind of real-world scenarios, the generated hypothesis described above obviously could not achieve the desired outcome. In order to simply the discussion, this paper focuses only on binary classification problems.

For the class imbalance problem, the degree of imbalance between classes may not be the only issue that hinders learning. Several research papers (He & Garcia, 2009) (Galar et al., 2012) have pointed out that data complexity would be the primary factor of classification performance deterioration, which is in fact intensified by the added skewed class distribution. More specifically, data complexity comprises the issues such as class overlapping (which makes discriminative rules hard to induce), lack of representative data, small disjuncts (which leads to underrepresented sub-concepts) and all of these issues contribute to performance degradation.

Over the past decades, several approaches have been proposed to address the challenges of imbalanced classification (Krawczyk, 2016). Some of them either aim to shift an inductive bias towards the positive class or apply a data preprocessing procedure to reduce the potentially undesirable impact of class imbalance on model building. Some other approaches assume higher misclassification costs for samples in the positive class and seek to minimize the higher misclassification errors in learning. Furthermore, several modifications or extensions of ensemble algorithms are recently adapted for imbalanced modeling, by embedding data preprocessing before applying each base learner or by integrating a cost-sensitive strategy in the ensemble learning process. We will discuss these different approaches in detail in the next section.

In this paper, we present a new hybrid learning framework called PRUSBoost for imbalanced classification. It applies a newly developed a partition-based data under-sampling strategy and integrates it into the AdaBoost algorithm (Freund & Schapire, 1996). It aims to provide a unified framework where we can informatively select some negative samples that exhibit mainstream characteristics of the class and also some negative samples that reveal significantly less typical features of the class and then combine them with the available positive samples to form a well-representative and balanced data set for training. This data selection process can be particularly helpful in the presence of data noises and class over-lapping regions in the data space. Once the training data samples are constructed, we further enhance the learning by building an ensemble of classifiers in hope of capturing most of the important underlying negative data patterns while learning most of the unique positive data features through an iterative process. The proposed framework can be considered as a general under-sampling approach that includes the well-known RUSBoost method (Steiffert et al., 2010) as a special case. Experiments on several data sets with various imbalance ratios indicate that the framework represents a very competitive and efficient alternative to handling imbalanced classification problems.

The rest of paper is organized as follows. Some closely related research work is reviewed in Section 2. The proposed hybrid learning framework for imbalanced classification is discussed in detail in Section 3. Experiments on several imbalanced data sets, including experimental setting, results and analysis, as well as a comparison with two popular hybrid boosting methods (RUSBoost and SMOTEBoost), are presented in Section 4. Finally, a few conclusion remarks are provided in Section 5.

## RELATED WORK

Over the past years, imbalanced learning has become a very popular research topic among the machine learning community. Several approaches have been proposed to address various challenges associated with imbalanced leaning and according to the study (Galar et al., 2012), they can be categorized into three primary groups: 1) algorithm level, 2) data level, and 3) cost sensitive.

The algorithm level approaches create new algorithms or modify existing ones and aim to alleviate the learning bias towards the negative class and to become more effective in learning imbalanced class distributions. For instance, decision tree algorithms can be adapted for imbalanced classification by adjusting the decision thresholds at leaf nodes, modifying the split criteria at each tree node, or by

developing more adequate pruning strategies. As another example, SVM can also be reworked for this purpose by using adequate penalty constants for different classes (Lin et al., 2002).

The data level approaches are those that apply a data preprocessing or data sampling step to reduce the negative effect of skewed class distributions in learning. There are multiple ways can be done for data sampling. The simplest option of balancing a training data set is to under-sample the negative class, which is typically accomplished randomly, while retaining all samples of the positive class. With a reduced training data set, these under-sampling methods help speed up the learning process, but it can result in information loss for the negative class. As an alternative to under-sampling, data rebalancing can be done by over-sampling the positive class by, for instance, simply duplicating positive samples. This approach, by contrast, does not lose any information of the positive class but, due to the resulted increased data size, it takes more time to build learning models. There are several more sophisticated over sampling options that have been proposed (Stefanowski & Wilk, 2008) and, among them, SMOTE (Chawla et al., 2002) is a popular one. SMOTE generates new synthetic positive samples randomly along the line segments connecting each available positive sample to its nearest neighbors and consequently could help find some unrevealed positive clusters. However, SMOTE can cause over generalization and potentially increase the occurrence of overlapping samples between classes (He & Garcia, 2009).

The cost-sensitive approaches assume there is a cost that is associated with misclassifying individual samples and in general, the cost of misclassifying a positive sample outweighs that of misclassifying a negative one. They seek to minimize the number of expensive errors and the total misclassification cost in the learning process (Elkan, 2001). One popular implementation of cost-sensitive learning is to assign samples of different classes with different weights, which are in proportion to their known or specified misclassification costs, and then use the weighted samples to build a classifier (Ting, 2002). This sample weighting technique places more focus on samples with higher weights and consequently it helps produce a more balanced classifier. A noticeable limitation with cost-sensitive learning, however, is that the required misclassification costs of a given application are either unknown or very difficult to be determined in practice.

Beside these three primary groups of imbalanced classification approaches, several ensemble algorithms have also been adapted as solutions to class imbalance problems (Lopez et al., 2013). Ensemble algorithms are primarily designed to maximize the accuracy criterion and hence they are not always very effective as direct solutions to imbalanced classification problems. However, over the recent years, several methods that integrate ensemble algorithms (boosting, bagging) into data sampling schemes (under-sampling, oversampling) have been proposed and their promising classification results have been reported. Among a few others, RUSBoost and SMOTEBoost (Chawla et al., 2003) are the most popular ones. Different from hybrid methods that combine data sampling schemes with non-ensemble classifiers, an ensemble is constructed by a sequence of individual classifiers, each of which is built on a different sampling of data, and the final combined classifier is expected to form a decision region that is more accurately reflected of training data. A report of related extensive experiments (Galar et al., 2012) have shown that ensemble-based hybrid approaches are among the most promising and efficient solutions for imbalanced classification.

## THE PROPOSED LEARNING FRAMEWORK (PRUSBOOST)

As discussed in the previous section, several ensemble-based hybrid methods have been proposed and evaluated on various experiments and the results have suggested that these methods are generally quite effective as solutions to imbalanced classification problems. Among them, RUSBoost is a representative one. It incorporates random under-sampling or RUS, a simple data rebalancing technique that randomly removes negative samples until the remaining ones match the samples from the positive class in size, into AdaBoost that iteratively builds a sequence of classifiers. Specifically, at each AdaBoost iteration, RUSBoost first applies RUS to form a balanced data set and then uses

it to build a new classifier that focuses more on those misclassified samples with the hope that they could be correctly classified during the next iteration. Once all classifiers are constructed, they are combined to form a final weighted classifier.

RUSBoost has a few benefits. It is simple and relatively fast in building learning models due to reduced data size. Although the random under-sampling scheme works reasonably well in general, it may lose some important information about the negative class. For instance, the selection can remove some critical negative samples that exhibit prevalent characteristics of the class. Further, it does not provide a mechanism of organizing samples into groups based on certain criteria and forming a balanced data set by selecting samples with certain proportions from the groups, which can potentially have an impact on classification performance.

In this section, we describe the proposed learning framework for imbalanced classification. Essentially, it applies a newly developed a partition-based data under-sampling strategy and integrates it into the AdaBoost algorithm. We name the framework PRUSBoost, or Partition-based RUSBoost. In comparison to RUSBoost, PRUSBoost aims to provide a unified framework where we can more informatively select certain negative samples that display mainstream features or show significant irregular characteristics of the class and, by combining them with available positive data, we build an ensemble of classifiers in hope of capturing most of important underlying data patterns through a boosting procedure. PRUSBoost can be considered a general under-sampling approach and in fact it includes RUSBoost as a special case.

There are a variety of different approaches that can be used to organize training samples of the negative class into groups based on certain criteria or characteristics. In this work, we limit ourselves to two sample groups and more specifically, we apply a novel detection algorithm to partition the negative sample population into 1) those demonstrate the prevalent or typical features of the class and 2) those display somewhat rare or irregular features of the class. The objective of this selective sampling process is to form a well-calibrated and reduced negative sample set that can, once combined with the given positive data, help find a reasonably good class decision boundary in the data space. A more simple and similar strategy for selecting samples with irregular behaviors has also been proposed for building models to identify thieves in public transit systems (Du et al., 2016).

In PRUSBoost, we use the one-class SVM algorithm (Scholkopf et al, 2001) to partition samples due to its solid theoretical foundation, efficient computations, and generally superior classification performance. Like the regular SVM, one-class SVM computes non-linear decision boundaries by using appropriate kernel functions and soft margins, but it constructs the boundaries that separate the data of the target class from the origin. Note that only a portion of the data points are allowed to be on the other side of the boundaries and these pointes are regarded as irregulars or outliers.

Assuming we have a set of training data with *n* samples

$$S = \{(x_1, y_1), ..., (x_n, y_n)\} \tag{1}$$

where $x_i$ is the vector in some space *X* that represents attribute values of a sample and $y_i \in [+1, -1]$ is the binary value indicating the sample's class label. Using a kernel function defined by $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, where the function $\phi(\bullet)$ transforms the data points from *X* to a high dimensional feature space *F*, the optimal decision boundary in the transformed space *F* can be expressed as

$$w^T \phi\left(x_i\right) - \rho = 0 \tag{2}$$

where $w$ is the vector in *F* perpendicular to the decision boundary and $\rho$ is the bias term. Further, the optimal decision boundary for one-class SVM can be obtained by solving the following optimization problem

$$min_{w,\rho,\xi_i} \frac{\| w \|^2}{2} + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

$$subject\ to\ w^T\phi\left(x_i\right) \geq \rho - \xi_i, \xi_i \geq 0, i = 1,2,\cdots,n \tag{3}$$

where $\xi_i$ is the slack variable to allow $x_i$ to be lie on the other side of the decision boundary and $\nu$ is the regularization parameter.

By using a kernel function, we can find the desired decision boundary without dealing with the exact form of the transformation function $\phi\left(\bullet\right)$. We use the well-known Gaussian kernel for our one-class SVM implementation as it guarantees the existence of such an optimal decision boundary separating data from the origin (Scholkopf et al., 2001).

For a given data set, we apply the one-class SVM algorithm to all samples of the negative class to partition them into the regular portion that contains the samples sharing some mainstream behaviors and the anomalous portion that contains the samples having relatively unique or unpopular behaviors. We can manipulate the regularization parameter $\nu$ to have a desired data split between the two portions. For instance, we may choose a data portion *p* (e.g., 20% or 30%) as anomalous and the remaining ((1 - *p*) %) as regular for the partition. Once the partition is formed, we can, based on our modelling needs, construct the final negative training data by randomly selecting a percentage *q* from the regular portion and the remaining ((1 - *q*) %) from the anomalous portion. The total number of selected samples is comparable or matched with the number of given positive samples (Weiss & Provost, 2003). Clearly, the partition between the two groups can be made with many different dividing choices and the optimal partition that leads to the best performed classifier is likely data dependent. In fact, the best partition might be associated with several factors of data samples such as imbalance ratio between the positive and negative classes, sample distributions and also the overall quality of the data.

Once a balanced training data set is formed, we integrate it into the boosting algorithm, or AdaBoost, to build a classification model. AdaBoost, or Adaptive Boosting, is a well-known ensemble machine learning algorithm and it can be used in conjunction with other base learning algorithms to improve classification performance. The method involves an iterative process that produces a sequence of classifiers or hypotheses and in each step of the process, it builds a classifier that focuses more on the training samples that are misclassified by the previous one. This is accomplished by using an adaptive weighting scheme on the training data. In the context of class imbalance problems, this unique weighting scheme of AdaBoost can be particularly helpful because the positive samples tend to be misclassified and, by assigning higher weights on them in subsequent iterations, we would expect to have an improved classification of the positive class.

The core algorithm of PRUSBoost is shown in Figure 1. In PRUSBoost, the boosting procedure takes as input a training data set (1) and applies a base learning algorithm repeatedly in multiple rounds to builds an ensemble of classifiers. It begins with an initial distribution $D_1$ of by assigning an equal weight to all training samples,

$$D_1\left(i\right) = w_i = 1/n,\ i = 1,2,\ldots,n$$

**Figure 1. The core algorithm of PRUSBoost**

Input: A set of labeled data $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in X$, $y_i \in \{-1, +1\}$
A maximum number of boosting iterations $T$
A base classification algorithm $C$
Partition parameters $p$ and $q$
Output: A classifier

Initialize the distribution of $S$: $D_1(i) = 1/n$, for $i = 1, 2, \ldots, n$
For $t = 1, 2, \ldots T$ do
1 Apply partition-based random under-sampling with parameters $p$ and $q$ on $S$ to form a training dataset $S_t'$ with a modified distribution $D_t'$
2 Run $C$ on $S_t'$ with $D_t'$ to produce a hypothesis $h_t: X \to \{-1, +1\}$
3 Compute the pseudo-loss of $h_t$ for $D_t$ and $S$
$\epsilon_t = \frac{1}{2}\sum_{(i,y):\, y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y))$
4 Compute the weight update term
$\beta_t = \epsilon_t / (1 - \epsilon_t)$
5 Update the distribution:
$D_{t+1}(i) = D_t(i) \times \beta_t^{\frac{1}{2}(1 + h_t(x_i, y_i) - h_t(x_i, y: y \neq y_i))}$
6 Normalize $D_{t+1}(i)$
Form and output the final hypothesis:

$$h_{final}(x) = \underset{y}{\text{augmax}} \sum_t (log \frac{1}{\beta_t}) h_t(x, y)$$

and on round *t,* it first applies the partition-based random under-sampling procedure with partition parameters *p* and *q* on $S$ to create a balanced training data set $S_t^{'}$ with an adjusted distribution $D_t^{'}$ (Step 1). Then, it runs the base algorithm to build a classifier or hypothesis (Step 2),

$$h_t\left(x,\ y\right): X \to \left\{-1,+1\right\} \tag{4}$$

Then, the boosting procedure computes the pseudo-loss of (x, y) with respect to the data set and the distribution (Step 3):

$$\varepsilon_t = \frac{1}{2} \sum_{(i,y):\, y_i \neq y} D_t\left(i\right)\left(1 - h_t\left(x_i, y_i\right) + h_t\left(x_i, y\right)\right)$$

and the sample weight update term (Step 4)

$$\beta_t = \varepsilon_t \,/\, (1 - \varepsilon_t) \tag{5}$$

which is used to update $D_t$ and to produce the final hypothesis. The distribution of $S$ for the next round, $D_{t+1}$, is then updated by (Step 5)

$$D_{t+1}(i) = D_t(i) \times \beta_t^{\frac{1}{2}(1+h_t(x_i,\ y_i)-h_t(x_i,\ y:y\neq y_i))}, i = 1,2,\ldots,n$$

Once all sample weights are updated, they are normalized (Step 6) and this normalization step effectively increases the weight for misclassified samples and decreases the weight for correctly classified ones. The process is repeated for several times to generate a sequence of classifiers. The final classification output from the procedure is formed by a weighted vote of the classifiers

$$h_{final}(x) = \underset{y}{\text{augmax}} \sum_t \left( log \frac{1}{\beta_t} \right) h_t(x,y) \tag{6}$$

where the weight for the individual classifier is given by $\log(1/\beta_t)$. This weight scheme gives a high weight to a classifier that performs well and gives a low weight to a classifier performs poorly.

## EXPERIMENTS AND ANALYSIS

### Data Sets

We evaluate the classification performance of PRUSBoost and compare it with two other state-of-the-art hybrid boosting methods (RUSBoost and SMOTEBoost) using twelve data sets gathered from a variety of application domains (Steiffert et al., 2010) that range from software defect prediction (CM1, PC1), health care (Mammography, Contr2), image recognition and analysis (SatImage4, Pendigits5, Segment5, Vehicle1), bioinformatics (Ecoli4) to vehicle evaluation (Car3) and material science (SolarFlareF, Glass3). All the sets are publicly available, and they can be accessed either from the UCI Machine Learning repository[1] or their pertinent websites. Table 1 shows the primary characteristics of the data sets.

For each of the sets, Table 1 includes the sample size, the number of attributes, the sample number of the positive class as well as the imbalance ratio (IR), which is defined as the count ratio of the negative samples to the positive samples. The included data sets have quite diverse characteristics and in particular, their IR values range from highly unbalanced 42.01 (Mammography) to moderately unbalanced 2.99 (Vehicle1), and their sample sizes spread from 214 (Glass3) to 11,183 (Mammography). The data sets are listed in Table 1 in the decreased order of IR.

**Table 1. Data sets used for experiments and their primary characteristics**

| Dataset | # of Samples | # of Attributes | # of Positive | Imbalance Ratio |
|---|---|---|---|---|
| Mammography | 11,183 | 7 | 260 | 42.01 |
| SolarFlareF | 1,389 | 13 | 51 | 26.24 |
| Car3 | 1,728 | 7 | 69 | 24.04 |
| PC1 | 1,109 | 22 | 77 | 13.40 |
| Glass3 | 214 | 10 | 17 | 11.59 |
| CM1 | 498 | 22 | 49 | 9.16 |
| Pendigits5 | 10,992 | 17 | 1,055 | 9.42 |
| SatImage4 | 6,345 | 37 | 626 | 9.28 |
| Ecoli4 | 336 | 8 | 35 | 8.60 |
| Segment5 | 2,310 | 20 | 330 | 6.00 |
| Contra2 | 1,473 | 10 | 333 | 3.42 |
| Vehicle1 | 846 | 19 | 212 | 2.99 |

The proposed framework PRUSBoost is not limited to the data with two classes and in fact it can classify unbalanced data with multiple classes in practice. But we only consider the binary classification problem in this paper. Hence, if an original data set included in the experiments has multiple classes, we select one of them as the positive or minority class and combine the rest to form the negative or majority class.

### Hybrid Boosting Methods Used for Comparison

To assess the effectiveness of PRUSBoost, we include two best known hybrid boosting-based methods to be compared with in the experiments, and they are RUSBoost and SMOTEBoost. As a special case of PRUSBoost, RUSBoost has been described in Section 3. These two approaches are closely related in terms of their algorithmic structures, and both apply a data sampling strategy to create a balanced training data set and then embed it into the AdaBoost iterations. The primary difference between these two is that RUSBoost uses a straightforward random under-sampling procedure while PRUSBoost applies a selective partition-based random under-sampling technique.

Unlike random oversampling, the SMOTE method creates new artificial samples for the positive class based on the feature space similarity between given positive samples. As a result of this process, SMOTEBoost can lead to a broadened decision region for the positive class. But it can also increase the occurrence of overlapping between classes and certainly result in larger leaning models due to its increased training data size.

### Experiment Setup

All three hybrid methods (RUSBoost, SMOTEBoost, PRUSBoost) create a balanced training data set for model building. The quality of the built models may depend on the produced training data. Further, there might be an optimal balance between the positive and negative classes in the sense that it yields the best performed classifier. Such a class balance is likely data dependent, and it is surely not easy to be determined. In this work, for both RUSBoost and PRUSBoost, we simply use under-sampling to form training data sets with an equal size between the two classes as this option is often considered to be near optimal (Weiss & Provost, 2003). For SMOTEBoost, we over-sample the positive class so that, together with original positive samples, it has a total matching the negative class. For the data partition used in PRUSBoost, we divide the negative class into a 20:80 ratio portions (i.e., $p = 20\%$) and then for each AdaBoost iteration, we randomly select a fixed 60% (i.e., $q = 60\%$) from the regular portion and the remaining 40% from the irregular one. Furthermore, the KEEL's implementations of RUSBoost and SMOTEBoost (Galar et al., 2012) are used in the experiments. In order to have a fair and consistent comparison, all three approaches use the well-known C4.5 decision tree algorithm (Quinlan, 1993) as the base learner for their boosting process and the process runs 10 iterations.

There are several metrics that can be used for assessing classification performance and guiding model learning. Accuracy is the most general metric for the purposes. However, it is not adequate for gauging solutions of class imbalance problems because the positive class has little impact on the accuracy metric compared to the negative class (Joshi et al., 2001). In other words, in these cases, accuracy reveals more about distribution of classes than it does about the actual performance of the models.

A popular and useful tool of assessing and comparing performance between different algorithms for class imbalance is the receiver operating characteristics (ROC) curve (Provost & Fawcett, 2001). For a given classifier, the ROC curve plots the true positive rate of a classifier on the vertical axis against its true negative rate on the horizontal axis. It also offers a nice visual interpretation of the tradeoffs between the benefits (reflected by true positives) and the costs (reflected by false positives) of classification regarding data distribution (He & Garcia, 2009). In order to compare different models with ROC, however, it is going to be hard to declare a winner unless one curve dominates the other(s) over the entire space. For this reason, the area under a ROC curve or AUC provides a functional

alternative, which is a singer performance measure for evaluating and determining which model is better on average, and it can be applied to imbalanced data sets.

In addition, precision and recall (or sensitivity) is a widely used pair of metrics to deliver a comprehensive assessment on class imbalance problems where successful classification of one class (the positive or minority class for instance) is considered more significant than that of the opposite class (the negative or majority class). For a classier, precision measures its exactness (the percentage of the samples classified as positive actually belong to the positive class) while recall or sensitivity measures its completeness (the percentage of the positive samples are classified correctly). A harmonic mean of precision and recall, or F1, is often used to gauge the classification effectiveness. Different from an arithmetic average, a high F1 value ensures that both precision and recall are reasonably high.

Furthermore, with sensitivity, there is a corresponding metric, called specificity, that can be used to measure the accuracy of the negative or majority class. It is the percentage of the negative samples classified correctly. Clearly, there is a trade-off between sensitivity and specificity values. The geometric mean (G-mean) of both metrics has also been a popular unified metric for imbalance class applications (Kubat et al., 1998). It is defined as

$$G - mean = \sqrt{sensitivity \times specificity}$$

and can be used to assess the balance of classification performance on both classes and in particular to help determine if a classifier overfits the negative class and underfits the positive class. A low G-mean value would typically indicate a poor performance in the classification of the positive class even if the negative class is mostly correctly classified.

For the experimental results reported in the next subsection, the metrics AUC, F1 and G-mean are used for classifier evaluation and comparison. All experiments in this work are conducted through the 5-fold stratified cross validation process. The presented results are the averaged values of the corresponding classification performance metrics.

## Experiment Results and Analysis

Table 2, 3 and 4 shows the classification performance results, measured by AUC, F1 and G-mean, respectively, obtained from the proposed framework PRUSBoost, along with RUSBoost and SMOTEBoost, on the twelve data sets included in the experiments. For each data set, the highest obtained matric value is highlighted in bold face. It can be observed from the tables that PRUSBoost achieves, even with simple and fixed settings on partition parameters for selecting negative samples, some very competitive and consistent classification performance, in comparison to the two state-of-the-art hybrid boosting approaches.

In order to have a quick view of how good an algorithm is with respect to others in comparison based on obtained tabular results, we compute the Friedman's average ranking for each of the three approaches over the data sets (Galar et al., 2012). Specifically, for each experimental setting that involves a data set and a performance assessment metric, we assign the top rank or rank 1 to the best-performed algorithm and then assign the next rank or rank 2 to the next followed-up algorithm, and so on. The final average ranking of an algorithm is computed by the mean value of its rankings over all data sets used in the experiments.

Figure 2 shows the combined average rankings of all three algorithms across all data sets and performance metrics, whereas Figure 3 shows the average rankings of the algorithms on an individual metric (i.e., AUC, F1 and G-mean). It can be seen from both figures that, among the comparison group, PRUSBoost has the lowest or top combined ranking and the lowest or top individual ranking on AUC and G-mean, while SMOTEBoost receives the lowest or top ranking on F1.

Friedman's average ranking provides an overview of the algorithms' performance considered in comparison. To show whether there are statistical differences among the achieved results, or to

**Table 2. AUC metric values for all data sets**

| Dataset | RUSBoost | SMOTEBoost | PRUSBoost |
|---|---|---|---|
| Mammography | **0.9434** | 0.9365 | 0.9397 |
| SolarFlareF | **0.8823** | 0.8550 | 0.8414 |
| Car3 | 0.9844 | **0.9979** | 0.9936 |
| PC1 | 0.8552 | 0.8600 | **0.8627** |
| Glass3 | 0.6670 | **0.8913** | 0.6574 |
| CM1 | 0.7308 | **0.7517** | 0.7282 |
| Pendigits5 | 0.9990 | 0.9988 | **0.9997** |
| SatImage4 | 0.9409 | **0.9539** | 0.9481 |
| Ecoli4 | 0.9244 | 0.9320 | **0.9347** |
| Segment5 | 0.9987 | 0.9993 | **1.0000** |
| Contra2 | 0.6983 | 0.6951 | **0.7119** |
| Vehicle1 | **0.8511** | 0.8445 | 0.8460 |
| Average | 0.8730 | 0.8930 | 0.8719 |

**Table 3. F1 metric values for all data sets**

| Dataset | RUSBoost | SMOTEBoost | PRUSBoost |
|---|---|---|---|
| Mammography | 0.3949 | **0.5973** | 0.3745 |
| SolarFlareF | 0.2544 | **0.3000** | 0.2262 |
| Car3 | 0.5188 | 0.5579 | **0.5823** |
| PC1 | 0.3438 | **0.4372** | 0.3377 |
| Glass3 | 0.1957 | **0.5000** | 0.2000 |
| CM1 | 0.2688 | 0.2742 | **0.2952** |
| Pendigits5 | 0.9524 | **0.9775** | 0.9725 |
| SatImage4 | 0.5450 | **0.6573** | 0.5508 |
| Ecoli4 | 0.5524 | **0.6353** | 0.5872 |
| Segment5 | 0.9821 | 0.9924 | **0.9940** |
| Contra2 | 0.4632 | 0.4428 | **0.4724** |
| Vehicle1 | 0.6316 | 0.5924 | **0.6452** |
| Average | 0.5086 | 0.5804 | 0.5198 |

**Table 4. G-mean metric values for all data sets**

| Dataset | RUSBoost | SMOTEBoost | PRUSBoost |
|---|---|---|---|
| Mammography | 0.8872 | 0.8707 | **0.9010** |
| SolarFlareF | **0.8303** | 0.7321 | 0.7795 |
| Car3 | 0.9606 | 0.9412 | **0.9697** |
| PC1 | 0.7869 | 0.6973 | **0.7997** |
| Glass3 | 0.5933 | **0.7710** | 0.5354 |
| CM1 | 0.6188 | 0.5497 | **0.6704** |
| Pendigits5 | 0.9884 | 0.9835 | **0.9924** |
| SatImage4 | 0.8706 | 0.8412 | **0.8763** |
| Ecoli4 | 0.8460 | 0.8747 | **0.8870** |
| Segment5 | 0.9970 | 0.9962 | **0.9990** |
| Contra2 | 0.6558 | 0.6302 | **0.6644** |
| Vehicle1 | 0.7724 | 0.7322 | **0.7917** |
| Average | 0.8173 | 0.8017 | 0.8222 |

**Figure 2. Average rankings on all combined metrics**

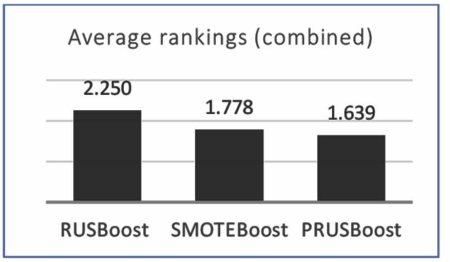
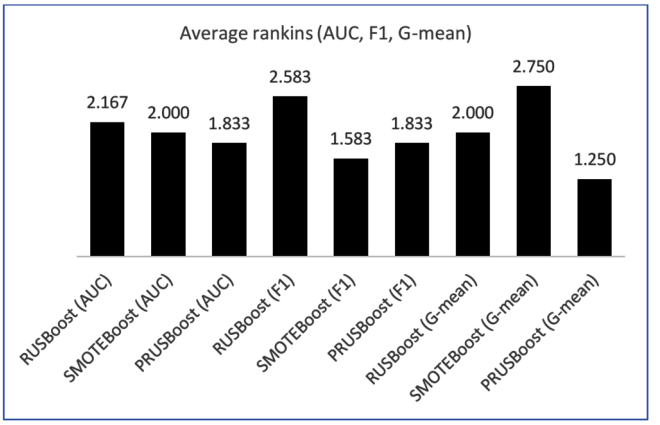



**Figure 3. Average rankings on individual metric: AUC, F1 and G-mean**




determine if there are one or more algorithms whose performance can be deemed as significantly superior or different, we need additional statistical tests. In contrast with parametric tests, proper non-parametric ones should be more adequate in this case as the data normality assumption required by typical parametric tests cannot be guaranteed. Specifically, for a comparison of multiple algorithms, the Iman-Davenport test (Iman & Davenport, 1980) can be applied to the results obtained by algorithms with different problems to detect if there are statistical differences among them. Furthermore, if there exist such differences, the Holm post-hoc test (Holm, 1979) can be followed up to reveal if a designated control algorithm is significantly better than the rest of algorithms in the comparison group (Garcia et al., 2009).

With the AUC metric, we can observe from Figure 3 that the rankings of all three algorithms are quite close to each other, although PRUSBoost leads the group in ranking. In other words, the absolute differences of the rankings are small. This is confirmed by the Iman-Davenport test with

**Table 5. Holm test result with F1**

| Algorithm | Z value | p-value | Holm | Hypothesis (.05) |
|---|---|---|---|---|
| RUSBoost | 2.4495 | 0.0143 | 0.025 | Rejected for SMOTEBoost |
| PRUSBoost | 0.6124 | 0.5403 | 0.05 | Not rejected |

**Table 6. Holm test result with G-mean**

| Algorithm | Z value | p-value | Holm | Hypothesis (.05) |
|---|---|---|---|---|
| SMOTEBoost | 3.6742 | 0.0002 | 0.025 | Rejected for PRUSBoost |
| RUSBoost | 1.8371 | 0.0662 | 0.05 | Not rejected |

a p-value of 0.7335. Therefore, in terms of AUC, all algorithms perform quite comparatively on the data sets. On the other hand, the differences on F1 among the algorithms seem to be relatively larger and SMOTEBoost leads the group. In fact, the corresponding Iman-Davenport test with a p-value of 0.0310 also indicates that the null hypothesis of all algorithms performing the same can be rejected. We then proceed with the Holm post-hoc test and the testing result are shown in Table 5. The Holm test result suggests that, as the control algorithm, SMOTEBoost is significantly better than RUSBoost with respect to F1. However, the same significance is not attained for PRUSBoost, which implies that although SMOTEBoost has a lower ranking than PRUSBoost, the difference between the two is not statistically significant. Finally, based on the performance results obtained on G-mean, the Iman-Davenport test produces a p-value of 0.0001 that indicates that we can safely reject the hypothesis of equivalence among the algorithms. In this case, PRUSBoost is used as the control algorithm because it has the lowest ranking in the group. The subsequent Holm post-hoc test implies that PRUSBoost performs significantly better than SMOTEBoost and, in comparison to RUSBoost, it is also significantly superior at a higher significance level (e.g. 0.1) but not at the default level of 0.05. The results are shown in Table 6.

Finally, we would like to provide two additional remarks on the comparison of the three algorithms. First, in contrast with RUSBoost, the proposed PRUSBoost framework is a hybrid boosting approach with a selective under-sampling scheme. In terms of computational cost, it requires a novelty model to partition samples of the negative class, but this additional workload should be reasonably low and once the model is built, it is used repeatedly in boosting iterations. On the other hand, SMOTEBoost involves a quite complex and expensive over-sampling process of generating synthetic samples for the positive class and additionally it takes more time to build a base learning model with an expanded data set. The time on model building is further amplified with the use of AdaBoost iterations. Second, each algorithm in the group has its own characteristics. A further analysis on produced confusion matrices of the algorithms indicates that the extra augmented positive samples of SMOTEBoost tend to shift its predictions towards the negative class and, as the result of this, it can lead to a relatively larger specificity and a smaller sensitivity (or recall) whereas the under-sampling counterparts, especially in the case of PRUSBoost, tend to improve the profile of the positive class and generate a larger recall and a smaller specificity. Clearly there are trade-offs between these two metrics, but it is plausible to argue that there are many real-world class imbalance problems where successful prediction of the positive class is considered more important and significant than prediction of the opposite class or classes. This desired preference is also consistent with the cost-sensitive learning approach where the misclassification cost of positives generally outweighs the misclassification cost of negatives.

## CONCLUSION

Imbalanced classification deals with the problems where the distribution of data samples across the given classes are skewed. This type of problems is ubiquitous in many areas of business and finance, engineering, medical research and health care, bioinformatics, system security and management, and so on. The uneven distribution in data, however, poses a significant challenge for accurate classification or prediction as most of the machine learning algorithms used for classification assume an equal

number of samples for each class. This results in models with poor performance especially for the class or classes with fewer samples.

We have presented a new hybrid learning framework named PRUSBoost as a solution to class imbalance problems. The framework under-samples the negative class through a novel partition-based procedure and integrates it into the well-known AdaBoost algorithm. The partition-based under-sampling procedure aims to select a well-representative balanced sample set for the negative class, which should be particularly helpful in the presence of data noises and class overlapping regions in the data space, and to facilitate a capable classifier to deal with a wide range of skewed class distributions.

We have applied the framework to a collection of data sets that are gathered from multiple diverse application domains that range from image recognition, health care, to bioinformatics and material science, and have a broad variety of class imbalance ratios. We also have compared the proposed PRUSBoost with the two state-of-the-art imbalanced classification methods, namely RUSBoost and SMOTEBoost. Experimental results and analysis have indicated that PRUSBoost represents a very competitive, consistent, and well-calibrated approach for imbalanced classification and is particularly effective in predicting samples of the positive or minority class, which is a desired outcome with many real-world applications.

We would like to continue and expand this imbalanced classification research work on several directions. One direction is to investigate possible optimal partition parameter settings with the framework PRUSBoost. For instance, we would like to find out what percentage of outlier samples should be used and what impact it may have on the overall classification performance. Another area in which we plan to explore is the use or development of other potentially suitable base learning algorithms for the framework.

## CONFLICT OF INTEREST

The author of this publication declares there is no conflict of interest.

## FUNDING AGENCY

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sections.

## REFERENCES

Akabani, R., Kwek, S., & Japkowicz, N. (2004). Apply support vector machines to imbalanced datasets. *Proceedings of European Conference on Machine Leaning*, 39-50.

Chawla, N. V., Hall, L. O., & Bowyer, K. (2003) SMOTEBoost: improving prediction of the minority class in boosting. *Proceedings of the Principles of Knowledge Discovery in Databases*, 107-119. doi:10.1007/978-3-540-39804-2_12

Chawla, N. V., Hall, L. O., Bowyer, K., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi:10.1613/jair.953

Du, B., Liu, C., Zhou, W., & Xiong, H. (2016) Catch me if you can: detecting pickpocket suspects from large-scale transit records. *Proceedings of 22nd International Conference on Knowledge Discovery and Data Mining*, 87-96. doi:10.1145/2939672.2939687

Elkan, C. (2001) The foundations of cost-sensitive learning. *Proceedings of 17th International Joint Conference on Artificial Intelligence*, 973-978.

Freund, Y., & Schapire, R. (1996) Experiments with a new boosting algorithm. *Proceedings of 13th International Conference on Machine Learning*, 148-156.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). 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*, *42*(4), 463–484. doi:10.1109/TSMCC.2011.2161285

Garcia, S., Fernandez, A., Luengo, L., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing*, *13*(10), 959–977. doi:10.1007/s00500-008-0392-y

He, H., & Garcia, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. doi:10.1109/TKDE.2008.239

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(20), 65–70.

Iman R. & Davenport, J. (1980). Approximations of the critical region of the Friedman statistic. *Journal of Communications in Statistics – Theory and Methods*, *9*(6), 571-595.

Joshi, M. V., Kumar, V., & Agawal, R. C. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. *Proceedings of IEEE international Conference on Data Mining*, 257-264. doi:10.1109/ICDM.2001.989527

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232. doi:10.1007/s13748-016-0094-0

Kubat, M., Holte, R., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, *30*(2/3), 195–215. doi:10.1023/A:1007452223027

Lin, Y., Lee, Y., & Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning*, *46*(1/3), 191–202. doi:10.1023/A:1012406528296

Lopez, V., Fernandez, A., Garcia, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113–141. doi:10.1016/j.ins.2013.07.007

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, *42*(3), 203–231. doi:10.1023/A:1007601015854

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Scholkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471. doi:10.1162/089976601750264965 PMID:11440593

Stefanowski, J., & Wilk, S. (2008). Selective pre-processing of imbalanced data for improving classification performance. *Proceedings of International Conference on Data Warehousing and Knowledge Discovery*, 283-292. doi:10.1007/978-3-540-85836-2_27

Steiffert, C., Khoshgoftaar, T., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics. Part A, Systems and Humans*, *40*(1), 185–197. doi:10.1109/TSMCA.2009.2029559

Ting, K. M. (2002). An instance weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, *14*(3), 659–665. doi:10.1109/TKDE.2002.1000348

Weiss, G., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, *19*, 315–254. doi:10.1613/jair.1199

## ENDNOTE

[1] UCI Machine Learning Repository https://archive.ics.uci.edu/ml/index.php

*Eric Jiang, Ph.D., is a full professor of computer science at University of San Diego. His research has been primarily in information retrieval, data analytics, machine learning, parallel and distributed computing. He has published a book "Nonlinear Numerical Analysis and Optimization" and research papers in journals and edited books. He has also given presentations at international conferences, workshops and technology research centers. Professor Jiang has been serving on journal editorial boards that include International Journal of Intelligent Data Analysis. In addition, he has served on organizing committees of numerous international conferences on information retrieval, machine learning, intelligent systems and applications.*