Roughly Balanced Bagging for Imbalanced Data

Shohei Hido* Hisashi Kashima*

Abstract

Imbalanced class problems appear in many real applications of classification learning. We propose a novel sampling method to improve bagging for data sets with skewed class distributions. In our new sampling method "Roughly Balanced Bagging" (RB Bagging), the number of samples in the largest and smallest classes are different, but they are effectively balanced when averaged over all subsets, which supports the approach of bagging in a more appropriate way. Our method is different from the existing bagging methods for imbalanced data which draw exactly the same numbers of majority and minority examples for the sampled subset data. In addition, our method makes full use of all of the minority examples by under-sampling, which is efficiently done by using negative binomial distributions. RB Bagging outperforms the existing "balanced" methods and other common methods, as shown by the experiments using benchmark and real-world data sets.

Keywords: bagging, imbalanced data, sampling, negative binomial distribution.

1 Introduction

In real world applications, we often encounter data sets whose class distribution is highly skewed. For example, the number of patients is much smaller than the number of healthy people in medical diagnosis data for normal populations. Again, the number of fraudulent actions is much smaller than that of normal transactions in credit card usage data. When a prediction model is trained on such an imbalanced data set, it tends to show a strong bias toward the majority class, since typical learning algorithms intend to maximize the overall prediction accuracy. In fact, if 95% of the entire data set belongs to the majority class, the model might ignore the remaining 5% of minority examples and predict that all of the test examples are in the majority class. Even though the accuracy will be 95%, the examples of the minority class will be absolutely misclassified. The misclassification cost for the minority class, however, is usually much higher than that of majority class and should not be ignored. In addition, the class distribution in a test set may be different from that of the imbalanced training set. In such cases the trained model will perform poorly on the test set.

To address this practically important problem, many studies have been conducted to improve learning algorithms for imbalanced data [5]. They mainly consist of re-sampling methods [4, 16], boosting-based algorithms [6, 15] and cost-sensitive ensemble approaches [9, 11, 22]. Bagging [2] has also been applied to classification problems with class imbalances. However, the only technique applied in the previous works is to correct the skewness of the class distribution in each sampled subset by using under-sampling or over-sampling. For example, the under-sampling methods typically sample a subset of the majority class data so that its size is equal to the size of the minority class data, and the methods use all of the data from the minority class, since every derived subset includes exactly the same number of majority and minority examples, the trained model performs equally for all classes. While such a strategy seems intuitive and reasonable at first sight, we claim that such existing strategies do not truly reflect the philosophy of bagging. Since the original bagging does bootstrap sampling from the whole data set independently of the class labels. the class distribution of each sampled subset varies according to the binomial distribution (or multinomial distribution for multi-class classification), and the averaged class distribution over all subsets tends to agree with the original class distribution. In contrast, in the existing bagging methods for imbalanced data, each subset has exactly the same class distribution as the desired (typically, uniform) distribution. Therefore, we can say that this allow us room to improve the performance of bagging for imbalanced data by using the philosophy of bagging in a more appropriate way. To the best of our knowledge, there is no previous work addressing this issue empirically or theoretically.

Our contributions in this paper are summarized as follows:

- A new under-sampling technique using a negative binomial distribution for bagging on imbalanced data.
- Justification of the proposed sampling method as an approximation of the original bagging.
- Extensive experimental evidence that RB Bagging works better with appropriate metrics.

The rest of this paper is organized as follows. Section 2

*IBM Research, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa, 242-8502 Japan. {hido, hkashima}@jp.ibm.com

provides the basis for the imbalanced class problem. We explain RB Bagging and its sampling technique in Section 3. We introduce the bagging-based algorithm to generate training subsets and to build an ensemble model. We address how to equalize the probability of choosing each class for a sample, rather than the sample size. The technique allows the class distributions of subsets to become slightly imbalanced and different. We discuss the interpretation of our sampling method as an approximation of bagging for the imbalanced class problem in Section 3.4. Section 4 describes the performance metrics and comparable learning algorithms for the evaluation. In Section 5, we investigate the performance of RB Bagging in experiments using benchmark and real-world data sets. Section 7 gives a brief summary and considers future work.

2 Imbalanced Class Problem

Real-world data set often has the problem of *class imbalance* or *skewed class distribution* in which the examples of the majority class outnumber those of the minority examples. This results not only from skewness in the class prior distributions, but from sampling bias. In such cases, the fraction of the minority class data can be 10% or even less. With such imbalanced training data sets, supervised classifiers face difficulty in the prediction of data with the minority class label. Since their purpose is typically to maximize the overall prediction accuracy, their predictions are strongly biased toward the majority class.

Table 1 shows an example of this problem as a confusion matrix. The rows show the sample sizes of actual classes in a test set and the columns represent the prediction by a classification model. In this case all majority examples are predicted correctly. Though the prediction accuracy is higher than 95%, most of the minority examples are misclassified as belonging to the majority.

We are focusing on such binary classification problems. We assume that the negative examples and the positive examples belong to the majority class and the minority class, respectively. Let us denote by x_i an input feature vector of d-dimensional real-valued or nominal-valued variables, by $y_i \in \{neg, pos\}$ the class label of x_i, and by $D = \{(x_1, y_1)...(x_n, y_n)\}$ the training data set whose class labels are highly skewed.

Table 1: Imbalanced confusion matrix

	Predicted negative	Predicted positive
Actual negative	True negative (TN) 400	False positive (FP) 0
Actual positive	False negative (FN) 20	True positive (TP) 4

In summary, our goal is defined as follows.

DEFINITION 2.1. (IMBALANCED CLASS PROBLEM) *Given the imbalanced data set D, estimate a prediction model that performs well when evaluated by performance metrics that balance the accuracies for both class labels.*

The "performance metrics" in the definition will be described in Section 4.1.

3 Roughly Balanced Bagging

3.1 Bagging for Imbalanced Data Bagging is one of the ensemble-based meta-learning algorithms which samples subsets from the training set, building multiple base learners and aggregating their predictions to make final predictions [2].

Let K denote the number of base learners. Data set D is converted into K equal-sized subsets $\{D^1, \cdots, D^K\}$ using bootstrap sampling. Let $f^k(x)$ be the base model trained on the k-th subset D^k, and $f^A(x)$ be the final ensemble model. The output of $f^A(x)$ is aggregated from the set of base models $\{f^1(x), f^2(x), \ldots, f^K(x)\}$. Though $f^A(x)$ is originally determined by voting of the predicted class labels, we use the average of the estimated probability $p^k(y|x)$ as follows.

$$f^A(x_i) = \frac{1}{K} \sum_{k=1}^{K} p^k(y_i|x_i)$$

This is because the averaged probability is known to result in higher prediction accuracies than label voting [10].

There are several studies which explain why bagging improves the predictive performance by reduction of the variance of the mean squared error. The amount of improvement depends on the bias-variance decomposition for base learners, which suggests that unstable models with high variances such as decision trees are preferable as the base learner for bagging rather than stable ones like logistic regression [2].

In contrast to the boosting-based algorithms [6, 15] for imbalanced data sets, bagging has been less attractive since its simple strategy leaves little space for handling class imbalance except changing the bagged size K and its subset sampling strategy. Tao et al. proposed a balanced sampling approach to perform bootstrap sampling only on the negative class so that its size is equal to the number of positive examples, and keep the entire positive examples in all subsets [20]. However, the original bagging chooses each bootstrap sample independently of the class labels, which results in the class distribution of each subset not being exactly the same as the original class distribution. Therefore it is unclear whether the aggregated model based on such exactly balanced subsets preserves the advantage of the original bagging or not.

- Inputs:
  D is the training data set
  L is the base learner
  K is the number of base learners
  x_i is an example drawn from the test set

- Build Roughly Balanced Bagging Model(D, L, K):
  Divide D into negative set D^{neg} and positive set D^{pos}
  For $k = 1$ to K
    Draw N_k^{neg} from the negative binomial distribution (3.1) with $n = N_k^{pos}$ and $q = 0.5$
    Set N_k^{pos} as the size of D^{pos} (i.e. $|D^{pos}|$)
    Let D_k^{neg} be N_k^{neg} examples sampled from D^{neg} with or without replacement
    Let D_k^{pos} be N_k^{pos} examples sampled from D^{pos} with or without replacement
    Build a model $f^k(x)$ by applying L to D_k^{neg} and D_k^{pos}
  Combine all $f^k(x)$ into the aggregated model $f^A(x)$
  Return $f^A(x)$

- Predict ($f^A(x)$, x_i, y):
  Calculate $p^A(y|x_i) = \frac{1}{K}\sum_{k=1}^{K} p^k(y|x_i)$ for all y
  Let $\hat{y} = \arg\max_{y \in \mathcal{L}} p^A(y|x_i)$
  Return $\hat{y}$

Figure 1: The RB Bagging Algorithm

3.2 Algorithm Based on the problem identified in the previous subsection, we aim to provide a natural extension of bagging for imbalanced data sets that reflects the philosophy of bagging in a more appropriate way. We believe the proposed approach will perform better in imbalanced data domains. The class distribution in the sampled subsets should be corrected to build base learners that can perform fairly for both of the classes. Also, we need to avoid information loss in the usual bootstrap sampling of rare positive examples.

We propose the *Roughly Balanced Bagging* (*RB Bagging* for short) to address these requirements. Figure 1 describes the algorithm. As with the original bagging, the input to the algorithm consists of a training dataset D, a base learner L, and a constant parameter K for the number of base learners. The important point is how to determine the numbers of samples for both classes here. We set the number of positive samples equal to that in the original dataset. If they are sampled without replacement, all of the positive examples will be contained in all of the sampled subsets. In contrast, the number of negative samples is decided probabilistically according to the *negative binomial distribution*, whose parameters are the number of minority (i.e. positive) examples and the probability of success $q = 0.5$, which will be discussed in detail later in Section 3.3. The key is that the examples of both classes are drawn with equal probability, but only the size of the negative samples vary, and the number of positive samples is kept constant since it is small. As reported by Friedman and Hall [13], there are no significant changes in results, and either sampling with replacement or sampling without replacement can be used. In prediction, the aggregated model simply outputs the class label with the highest average probability estimated by the base models.

3.3 Negative Binomial Sampling When a subset is chosen by bootstrap sampling over two equally-sized datasets, one of which belongs to the positive class and the other to the negative class, then the class distribution in the resultant subset must vary, but it also follows the binomial distribution with probability $q = 0.5$. When we are given an imbalanced class dataset, to sample an equally-balanced subset, we choose each class with probability 0.5, and draw a sample uniformly at random from the dataset belonging to the chosen class, and repeat this procedure until the size of the sampled subset reaches some fixed size.

However, this sampling method cannot control the size of the minority samples. Also, we would like to use the under-sampling to make sure to use (almost) all of the minority class data. Therefore we use the negative binomial distribution to achieve this purpose. The negative binomial distribution is a probability distribution of the number of failures m in Bernoulli trials given the number of successes n, which is defined by the following probability mass function:

$$p(m|n) = \begin{pmatrix} m+n-1 \\ n \end{pmatrix} q^n (1-q)^m \tag{3.1}$$

where p is the probability of success. For our purpose, we set $q = 0.5$. Note that the negative binomial distribution with integer parameter n is also called Pascal distribution though the parameter can be positive real number in general.

Figure 2 shows the distribution of m with $q = 0.5$

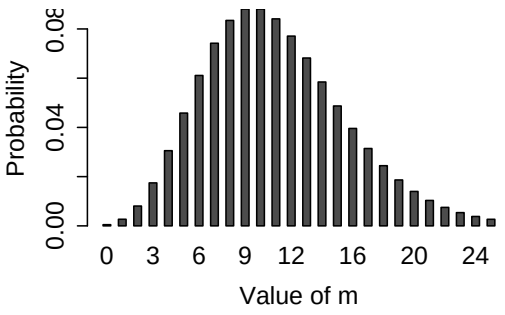

Figure 2: Distribution of the negative binomial distribution ($q = 0.5, n = 10$)

and $n = 10$. We can observe that m falls around $n = 10$ with high probability. When we use sampling without replacement, there is a small chance of the sample size m being larger than the number of data objects of the majority class. In such a case, we simply set $m = |D^{neg}|$.

3.4 Justification of the Proposed Sampling Method Our sampling method essentially depends on the widely used concept called *under-sampling*, which tries to make full use of minority class examples. At the same time, we also aim to naturally extend the original bagging to handle the imbalanced class problem by subset sampling.

Let us review the details of subset sampling in the original bagging. Basically, its bootstrap sampling (sampling with replacement) draws an i.i.d sample from the empirical joint distribution $p(x, y)$ a fixed number of times N (usually, $N = |D|$). The point is that the bootstrap sampling can be separated into two steps using decomposition $p(x, y) = p(x|y)p(y)$. First we determine the sample size of each class according to the prior probability $p(y)$. In fact, the sample size of each class follows a binomial distribution. For a balanced class distribution ($p(y) = 0.5$ for both y) then the density distribution of sample size $|D_y^k|$ is:

$$p(|D_y^k| = m|M) = \left(\begin{array}{c} M \\ m \end{array} \right) 0.5^M \tag{3.2}$$

Second we draw $|D_y^k|$ samples as determined by the conditional probability $p(x|y)$, which is done by taking $|D_y^k|$ samples uniformly at random from the examples belonging to the class y.

In order to address the imbalanced class problem, we perform sampling under the balanced class distribution. The simplest way to do this is to use the balanced prior $p'(y = neg) = p'(y = pos) = 0.5$ instead of the true $p(y)$ in the first step. Then the sample size of each class varies according to the balanced binomial distribution (3.2), where the number of positive samples and that of negative samples are equal on average. This is the most natural extension of the original bagging to cope with class imbalance.

However, strangely enough, none of the previous work adapts the bootstrap sampling in this way. Since the simple implementation described above cannot guarantee the size of the positive examples in each subset, some of models might contain only so small a fraction of the positive examples that the base model works poorly for the positive class. Instead, the existing algorithms fix the sample size, and draw the same number of samples from both classes for each subset. This means that the number of samples from each class is replaced by its expected value instead of sampling the number of samples itself. Though such an exactly balanced sampling method performs better for imbalanced data set than the original bagging, it seems to be still an open problem to explain why it works. The difficulty we face is that there is a conflict between the bootstrap sampling and the under-sampling. To the best of our knowledge, there is no such extension that satisfies both requirements simultaneously.

To resolve the conflict, we ease the restriction of the equal-size subsets. Our sampling strategy can be interpreted as repeating instance-by-instance sampling that selects the sampled class based on $p'(y) = 0.5$ and draws one example from the class until the total size of the sampled positive examples reaches the size of all of the positive examples. Using this technique we can implement the balanced sampling using $p'(y) = 0.5$ and make full use of the positive examples at the same time. Actually, the method is equivalent to bootstrap sampling where the sample size of each class is selected according to a negative binomial distribution with $p'(y) = 0.5$. Though the size of the subsets differs slightly from subset to subset, they are almost balanced on average, which is the same as the case of the original bagging.

Based on the above discussion, we can say that, by using the negative binomial sampling, the RB Bagging both preserves the nature of the original bagging and makes effective use of the information of all of the minority class data. We will evaluate the effectiveness of the proposed approach more precisely in Section 5.

4 Evaluation

In this section, we review several performance metrics to evaluate learning algorithms on imbalanced data sets. Then we review several algorithms that we will use as the base learner and comparison methods in the experiment.

4.1 Performance Metrics We review seven performance metrics commonly used for the evaluation of methods for the imbalanced class problem. Let us denote by D_{test} the test set ($N_{test} = |D_{test}|$). We assume that the prediction results are given in the same form as Table 1.

Prediction Accuracy.
Prediction accuracy represents the population of the correctly predicted examples. In the case of imbalanced class problem, as mentioned in Section 2, an overly simple model which predicts all test examples as the negative class might maximize the accuracy. Therefore we do not put so much weight on the naive accuracy in this paper.

$$Accuracy = \frac{TN + TP}{TN + FN + TP + FP}$$

F-measure.
The F-measure combines Precision $TP/(TP + FP)$ and Recall $TP/(TP + FN)$ on the prediction of positive class. A higher F-measure value indicates that the model performs better on positive class balancing of FP and FN.

$$F{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

G-mean.
The G-mean is calculated as the product of the prediction accuracies for both classes. Even if a model classifies the negative examples correctly, a poor performance in prediction of the positive examples will lead to a low G-mean value. In fact, the G-mean is quite important to measure the avoidance of the overfitting to the negative class and the degree to which the positive class is ignored.

$$G{-}mean = \sqrt{\frac{TN}{(TN+FP)} \times \frac{TP}{(TP+FN)}}$$

Mean Squared Error.
The Mean Squared Error (MSE) shows the error of the estimated probability of the actual class label y_i. A model which gives a precise probability estimation can reduce the MSE. Though it uses the true posterior probability $p^T(y_i|x_i)$ as the answer by original definition, we usually do not know $p^T(y_i|x_i)$ in real-world data sets. Instead, in general MSE is empirically calculated assuming $p^T(y_i|x_i) = 1$.

$$MSE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (1 - p(y_i|x_i))^2$$

Improved Squared Error.
The MSE is commonly used as a metric of estimated probability. Even if the model predicts all class labels correctly, however, the MSE will still becomes high if the probability estimation is unstable. Alternatively, Fan et al. introduced the Improved Squared Error (ISE) [10]. ISE accounts for the probability error only if the model classifies the example incorrectly. In other words, the ISE combines the accuracy and the MSE into one value. Note that we assume the prediction threshold of the estimated probability in the binary classification is fixed as 0.5.

$$ISE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (1 - min(1.0, \frac{p(y_i|x_i)}{0.5}))^2$$

ROC Curve and AUC.
From the early stages of research on imbalanced data, The ROC curves have been the primary metric to evaluate the performance of algorithms. The curves represent the trade-off between the TP and FP and the upper curve corresponds to better performance in learning from imbalanced data. The Area Under ROC Curve (AUC) is a value which indicates the extent of the area below a ROC curve. In contrast to the difficulty of identifying clear advantages in multiple crossed ROC curves, the order of the AUC values on the classification results is proved to be equivalent to the order of their significances in the statistical tests including Wilcoxon's test [1]. Therefore, we focus on the value of AUC as a metric rather than the complicated ROC curves.

4.2 Algorithms for Imbalanced Data Even for imbalanced class problems, decision tree algorithms, especially C4.5 [18], are the most widely used approaches. It is well-known that bagging with tree algorithms is a good idea since the aggregation is effective against the instability of trees by reducing the variance in the MSE [2, 10]. In contrast, more stable models such as logistic regression are not appropriate as the base learner. For this reason, we use C4.5 as the base learner for the RB Bagging and other ensemble methods in Section 5.

Following the previous work [6, 15], we used AdaBoost [12] and RIPPER [8] as the comparison algorithms in the experiments. Boosting is a family of ensemble algorithms that assigns larger weights to the misclassified examples from the current base model and trains the next base model using the new weights. If the base learners are strongly biased toward negative examples, then boosting can revise the weights of the misclassified positive examples and automatically cope with the imbalanced class problem. AdaBoost [12] is the most successful boosting algorithm. RIPPER is a rule based algorithm that generates rule sets using an MDL-based (Minimum Description Length) stopping criteria and does greedy pruning of the rules to minimize their description length. Since rules are generated for each class, RIPPER also performs well for imbalanced data sets.

5 Experiment

5.1 Data Sets and Experimental Settings We summarize the characteristics of the nine data sets including benchmark and real-world data set evaluated in our experiments in Table 2. Following the earlier studies [1, 6, 15], we randomly choose eight frequently used benchmark data sets from the UCI repository [17]. The multi-class data sets were converted into binary data sets which include the minority labels shown in the "Min. Label" column. The examples of the other classes are assigned the majority label. The largest data set has 20,000 items and the ratio of the smallest minority class ranged from 34.77% to 3.95%. Note that

Table 2: Summary of the imbalanced data sets

Data set	Size	#Attr.	Min. label	%Minority
Diabetes	768	8	pos	34.77%
Breast	286	9	malignant	34.48%
German	1000	20	bad	30.00%
E-Coli-4	336	7	iMU	10.42%
Satimage	6435	36	4	9.73%
Flag	194	28	white	8.76%
Glass	768	9	6	7.94%
*RealF	6651	77	low	4.99%
Letter-A	20000	16	a	3.95%

RealF is the real-world data set that we were given by a financial company.

We compared two models based on the proposed algorithm with seven models based on the other algorithms. We tested RB Bagging with size $K = 100$, with and without replacement, using C4.5 as the base learner. We also implemented the Exactly Balanced Bagging and Breiman's original bagging of equal size ensembles. For the usual models and base learners, we employed the widely used data mining tool *Weka* [21]. C4.5, AdaBoost and RIPPER are implemented as J48, AdaBoostM1 and JRip respectively in Weka. The single C4.5 tree is constructed to see the difficulties of learning on the data sets. The following two models are AdaBoost with size $K = 100$ and 200 (number of iterations). Their base learners were also C4.5. As a rule-based algorithm, we made use of two models based on RIPPER with the numbers of optimizations $Optimize$ set to 2 and 10. The other parameters are all the same as the default parameters in Weka.

All of the experiments and the statistical processing were performed in the statistical language environment *R* [19]. In the following experiments, we ran ten-fold cross validation in a stratified manner so that the training set and test sets preserve the same class distributions as the original sets. Then we calculated six metrics: AUC, MSE, ISE, F-measure, G-mean, and Accuracy as taken from the performance metrics introduced in Section 4.1. The value of each metric is averaged over ten trials.

5.2 Benchmark Data Sets We summarized the performance metrics resulting from the eight models for the benchmark data sets in Table 3 and Table 4.

Surprisingly, on the Diabetes, German and Satimage data sets, RB Bagging consistently outperformed Exact Balanced Bagging for all of the metrics. The results suggests our approach to make use of the roughly balanced subset is promising. Our algorithm also worked better compared to the usual algorithms including original bagging. The differences are clear in AUC and the G-mean.

The Flag data set shows that the values of the F-measure and G-mean with original bagging, C4.5, and AdaBoost are zero. These extreme observations show that they failed completely in the prediction of positive examples in every trial. In contrast, RB Bagging showed well-balanced performance for this difficult data set especially for AUC and the G-mean. While RIPPER had higher accuracy than RB Bagging, it seems to be slightly overfitted toward the correct classification of negative examples.

On the Glass data set, RB Bagging had the same accuracy as AdaBoost and outperformed it in the other metrics except for MSE.

The Letter-A data set indicates that AdaBoost worked nearly perfectly and outperformed RB Bagging in all of the metrics except for AUC. The reason would be that this problem is so easy that a single C4.5 tree can provide acceptably high performance. We conclude that it is difficult to make the ensemble model based on RB Bagging converge to the Bayesian error in these simple cases.

Let us review the performance for each metric. AUC is the place RB Bagging worked best, especially on Diabetes, German, and Flag. Interestingly, RB Bagging always resulted in a quite low ISE value even if it has a larger MSE than others. This indicates that RB Bagging tends to fail only if it becomes less confident and the estimated probability $p(y_i|x_i)$ is around 0.5. The advantages of RB Bagging for the MSE and F-measure are unstable and less clear for all algorithms. As expected, RB Bagging showed lower prediction accuracy for most data sets due to the incorrect classification of negative examples. Breiman's original bagging does not outperform RB Bagging for AUC except for the Letter-A data set, though it consistently works well for the MSE and prediction accuracy.

5.3 Real-world Data Set The RealF examples consist of eleven processed features of the customers for seven months in the loan business. Though we cannot make the data set public due to privacy and confidentiality issues, it is definitely valuable to evaluate the prediction algorithms for imbalanced data with a real-world application.

The motivation in analyzing this data set is to estimate the condition of customers in order to more quickly stop doing business with bad customers, since the increase of uncollectable debts has a large impact on the profits of such companies. The class label is the risk of a customer, low or high, determined after six months. In this data set, there is a clear difference between the performance of RB Bagging and the others especially for AUC. The relative performances of the algorithms are identical compared to those of the benchmark data sets. By putting importance on the high risk customers, only RB Bagging had well balanced AUC and G-mean values. In contrast, the other algorithms showed strong bias toward the prediction of the low risk (i.e. majority) customers. Therefore, RB Bagging can be used to detect untrusted customers and to reduce the debts. This clear advantage assures that our method is also promising for real-world applications.

In summary, RB Bagging almost always outperforms Exact Balanced Bagging by all of the metrics. It usually worked better than the other algorithms for AUC, ISE and G-mean. The values of MSE and F-measure were also comparable. As mentioned in Section 4.1, the trade-off between accuracy and emphasizing the minority is basically unavoidable. Seeing the overall accuracy as the balancing factor, the accuracy of RB Bagging seems more acceptable than that of Exact Balanced Bagging.

Table 3: Statistics of the experimental results on benchmark data sets (taken from the UCI repository)

Data set	Algorithm	AUC	MSE	ISE	F-measure	G-mean	Accuracy
Diabetes	RB Bagging (K=100)	83.5	0.162	0.0490	69.2	76.2	76.3
	RB Bagging w/ replace (K=100)	83.7	0.161	0.0487	70.4	77.2	77.5
	Exact Balanced Bagging (K=100)	82.7	0.173	0.0541	67.5	74.7	74.1
	Original Bagging (K=100)	83.4	0.157	0.0513	63.8	71.2	76.4
	C4.5 (pruned)	76.6	0.193	0.106	59.0	67.6	73.3
	AdaBoost (K=100)	79.6	0.244	0.239	63.7	71.4	75.1
	AdaBoost (K=200)	78.9	0.255	0.251	63.0	70.9	74.1
	RIPPER (Optimize=2)	69.5	0.193	0.0854	58.6	67.1	73.6
	RIPPER (Optimize=10)	71.2	0.188	0.0885	61.5	69.5	74.9
Breast	RB Bagging (K=100)	98.8	0.0359	0.0182	94.1	95.7	95.8
	RB Bagging w/ replace (K=100)	98.7	0.0358	0.0181	94.1	95.7	95.8
	Exact Balanced Bagging (K=100)	98.6	0.0413	0.0257	93.1	95.1	95.1
	Original Bagging (K=100)	98.3	0.0374	0.0198	93.7	95.4	95.6
	C4.5 (pruned)	96.4	0.0467	0.0332	92.2	94.1	94.6
	AdaBoost (K=100)	98.3	0.0293	0.0288	95.7	97.0	97.0
	AdaBoost (K=200)	98.2	0.0301	0.0301	95.7	97.0	97.0
	RIPPER (Optimize=2)	92.7	0.0619	0.0525	89.6	91.5	93.0
	RIPPER (Optimize=10)	92.9	0.0574	0.0522	90.8	92.3	93.8
German	RB Bagging (K=100)	77.3	0.188	0.0431	77.7	70.1	71.1
	RB Bagging w/ replace (K=100)	78.1	0.186	0.0415	77.3	69.9	70.8
	Exact Balanced Bagging (K=100)	76.0	0.208	0.0566	71.8	67.9	65.9
	Original Bagging (K=100)	76.9	0.173	0.0566	82.5	60.4	74.0
	C4.5 (pruned)	66.2	0.227	0.146	80.2	56.3	70.8
	AdaBoost (K=100)	71.0	0.249	0.245	83.0	62.6	74.9
	AdaBoost (K=200)	70.0	0.249	0.248	82.9	63.4	75.0
	RIPPER (Optimize=2)	63.5	0.194	0.0723	81.5	58.4	72.6
	RIPPER (Optimize=10)	63.9	0.197	0.0758	80.6	59.9	71.8
E-Coli-4	RB Bagging (K=100)	94.7	0.0877	0.0365	62.7	89.3	87.5
	RB Bagging w/ replace (K=100)	95.7	0.0871	0.0365	61.2	88.9	86.9
	Exact Balanced Bagging (K=100)	94.0	0.103	0.0516	58.5	88.3	85.7
	Original Bagging (K=100)	94.3	0.0460	0.0215	65.3	74.1	93.8
	C4.5 (pruned)	81.7	0.0523	0.0449	63.7	69.5	94.4
	AdaBoost (K=100)	93.7	0.0680	0.0679	62.2	70.1	93.2
	AdaBoost (K=200)	93.3	0.0775	0.0756	55.4	65.7	92.0
	RIPPER (Optimize=2)	77.1	0.0619	0.0478	57.3	67.8	92.6
	RIPPER (Optimize=10)	78.8	0.0671	0.0513	61.6	74.7	91.7

Table 4: Continuation of Table 3; Statistics of the experimental results on benchmark data sets (taken from the UCI repository)

Data set	Algorithm	AUC	MSE	ISE	F-measure	G-mean	Accuracy
Satimage	RB Bagging (K=100)	95.4	0.0785	0.0243	60.5	87.6	89.0
	RB Bagging w/ replace (K=100)	95.5	0.0781	0.0235	60.0	87.5	88.8
	Exact Balanced Bagging (K=100)	95.4	0.0960	0.0344	56.0	88.1	86.0
	Original Bagging (K=100)	95.5	0.0424	0.0159	65.9	73.9	94.4
	C4.5 (pruned)	76.1	0.0751	0.0705	57.4	72.5	92.1
	AdaBoost (K=100)	96.7	0.0495	0.0492	70.1	76.9	95.0
	AdaBoost (K=200)	96.8	0.0503	0.0501	69.5	76.5	94.9
	RIPPER (Optimize=2)	74.7	0.0636	0.0481	56.8	70.9	92.3
	RIPPER (Optimize=10)	75.8	0.0640	0.0501	58.1	72.4	92.4
Flag	RB Bagging (K=100)	75.2	0.178	0.0197	25.8	55.4	72.1
	RB Bagging w/ replace (K=100)	74.5	0.179	0.0200	21.3	47.4	71.1
	Exact Balanced Bagging (K=100)	74.2	0.212	0.0555	22.9	54.0	62.1
	Original Bagging (K=100)	61.0	0.0795	0.0600	0.0	0.0	91.3
	C4.5 (pruned)	50.0	0.0792	0.0591	0.0	0.0	91.3
	AdaBoost (K=100)	67.2	0.0817	0.0574	0.0	0.0	91.3
	AdaBoost (K=200)	67.2	0.0817	0.0574	0.0	0.0	91.3
	RIPPER (Optimize=2)	61.5	0.0854	0.0566	19.7	23.7	88.6
	RIPPER (Optimize=10)	64.8	0.0884	0.0596	28.0	30.9	88.5
Glass	RB Bagging (K=100)	96.7	0.0466	0.0231	85.9	92.9	95.8
	RB Bagging w/ replace (K=100)	96.6	0.0474	0.0240	86.7	92.8	95.8
	Exact Balanced Bagging (K=100)	95.4	0.0495	0.0257	85.3	92.5	95.3
	Original Bagging (K=100)	93.3	0.0368	0.0280	83.3	87.9	95.3
	C4.5 (pruned)	93.6	0.0415	0.0371	84.0	89.5	95.3
	AdaBoost (K=100)	95.2	0.0418	0.0415	84.8	89.8	95.8
	AdaBoost (K=200)	95.2	0.0418	0.0415	84.8	89.8	95.8
	RIPPER (Optimize=2)	91.2	0.0373	0.0341	85.5	90.5	96.2
	RIPPER (Optimize=10)	89.6	0.0417	0.0385	83.5	88.7	95.8
Letter-A	RB Bagging (K=100)	99.9	0.0103	0.00207	99.4	98.7	98.9
	RB Bagging w/ replace (K=100)	99.9	0.0103	0.00219	99.4	98.7	98.9
	Exact Balanced Bagging (K=100)	99.9	0.0128	0.00300	99.2	98.6	98.4
	Original Bagging (K=100)	100	0.00210	0.00100	99.9	97.4	99.7
	C4.5 (pruned)	98.9	0.00330	0.00305	99.8	97.7	99.6
	AdaBoost (K=100)	99.4	0.000557	0.000550	100	99.3	99.9
	AdaBoost (K=200)	99.4	0.000550	0.000550	100	99.3	99.9
	RIPPER (Optimize=2)	97.9	0.00334	0.00292	99.8	97.8	99.6
	RIPPER (Optimize=10)	98.0	0.00311	0.00296	99.8	97.9	99.7

Table 5: Statistics of the experimental result on the real-world data set (RealF)

Data set	Algorithm	AUC	MSE	ISE	F-measure	G-mean	Accuracy
RealF	RB Bagging (K=100)	83.4	0.112	0.0136	92.5	72.4	86.4
	RB Bagging w/ replace (K=100)	83.4	0.112	0.0140	92.3	72.8	86.2
	Exact Balanced Bagging (K=100)	82.5	0.157	0.0275	85.4	72.8	75.5
	Original Bagging (K=100)	81.9	0.0321	0.0260	98.2	56.4	96.5
	C4.5 (pruned)	65.6	0.0342	0.0308	98.2	55.4	96.4
	AdaBoost (K=100)	67.9	0.0332	0.0332	98.3	58.5	96.7
	AdaBoost (K=200)	67.8	0.0329	0.0329	98.3	58.8	96.7
	RIPPER (Optimize=2)	62.3	0.0367	0.0329	98.0	49.1	96.2
	RIPPER (Optimize=10)	64.0	0.0352	0.0316	98.1	52.4	96.3

6 Related Work

A traditional approach for the imbalanced class problem is intelligent re-sampling. Kubat and Martin proposed one-sided selection (OSS) based on the under-sampling of the majority examples which lie around the possible borderline or noisy area [16]. SMOTE is an over-sampling technique [4]. The method generates some synthetic minority examples by interpolating the minority examples carefully so that it avoids overfit. Their goal, which is to choose the best subset for a single model, is different from our approach to obtain better multiple subsets for an ensemble model.

Boosting could solve the imbalanced class problem naturally since it automatically assigns greater weights to the minority examples. In this paper, we compare RB Bagging with the most successful boosting algorithm, AdaBoost [12]. Recently, boosting algorithms using over-sampling techniques for class imbalance were presented. SMOTEBoost [6] applies SMOTE-based over-sampling to change the total weights of the misclassified minority examples. While the authors presented favorable experimental results for the metrics of precision, recall, and F-measure, it still remains a non-trivial problem to determine what amount of over sampling is enough. DataBoostIM [15] tries by generating data to balance not only the class distribution but also the total weight within the classes. The broad empirical studies show that DataBoostIM generally outperforms other algorithms including SMOTEBoost. However, there is no evaluation in terms of AUC, which is the largest advantage of RB Bagging, on the boosted over-sampling methods. In addition, RB Bagging does not require synthetic examples which may result in artifacts.

Cost-sensitive algorithms intend to minimize the total cost of the misclassification when the costs are given. There are many cost-sensitive algorithms based on the ensemble methods, such as MetaCost [9], Costing [22] and AdaCost [11]. Based on a bagging-like ensemble, MetaCost makes any base learner cost-sensitive. Costing applies cost-proportionate rejection sampling, but also generates subsets with slightly imbalanced total-cost-within-class. While these algorithms can handle the class imbalances with high misclassification costs for the minority class, empirical comparisons are impossible, since the objectives and performance metrics are different from ours.

Breiman performed some experiments using the same data sets from the UCI repository as used in our experiments [2]. He simply duplicated the minority examples and handled the original bagging so that the multiple replications of an example in a subset may lead to overfit. In fact, we re-implemented those experiments and found that it never outperforms RB Bagging. Gao et al. recently proposed an ensemble-based framework for a data stream with a skewed distribution [14]. Chan et al. extended Random Forest [3] to learn from imbalanced data by performing bootstrap sampling only on the minority class [7]. Since they also draw exactly the same number of minority and majority examples, our negative binomial sampling might improve the algorithm, using the slightly imbalanced subsets.

7 Conclusion

We addressed the imbalanced class problem from practical perspectives. Though many ensemble-based algorithms have been proposed for the problem, bagging has not been used that often. In this paper, we proposed Roughly Balanced Bagging (RB Bagging) to equalize the sampling probability of each class, instead of fixing the sample size as a constant. The size of the majority examples is determined probabilistically according to the negative binomial distribution. The class distribution of the sampled subsets becomes slightly imbalanced as well as the original bagging for balanced data sets. RB Bagging is successful in both preserving the nature of the original bagging, and at the same time making effective use of the information of all of the minority examples. This aggregated model becomes more robust than the common approach depending on exactly balanced subsets.

We evaluated our algorithm in the experiments using nine data sets including the usual benchmark data sets and

a real-world data set. We compared it with the exactly balanced model and other well-known algorithms such as AdaBoost and RIPPER for imbalanced data. RB Bagging generally outperformed them, especially for the performance metrics such as AUC, ISE, or G-mean known to be appropriate for the imbalanced class problem. For the real-world financial data set, RB Bagging showed a very clear advantage. These results show that our approach is practical and promising.

There are two areas for future work. The first one is more theoretical analysis to clarify the reason why RB Bagging works better than the other methods. The second one would be to perform more experiments, especially in comparison to the boosting-based algorithms with over-sampling such as SMOTEBoost and DataBoostIM. Since their data generation approach and our negative binomial sampling are not mutually exclusive, it would also be interesting to combine these techniques.

References

[1] G. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence and Research*, 16:321–357, 2002.

[5] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[6] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 107–119, 2003.

[7] C. Chen, A. Liaw, and L. Breiman. Using random forest to learn imbalanced data. Technical report, Department of Statistics, University of California, Berkeley, 2004.

[8] W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 115–123, 1995.

[9] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[10] W. Fan, E. Greengrass, J. McCloskey, P. S. Yu, and K. Drummey. Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM)*, pages 154–161, 2005.

[11] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proceedings of the Sixth International Conference on Machine Learning (ICML)*, pages 97–105, 1999.

[12] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[13] J. Friedman and P. Hall. On bagging and nonlinear estimation, available at *http://www-stat.stanford.edu/˜jhf/* , 2000.

[14] J. Gao, W. Fan, J. Han, and P. S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the Seventh SIAM International Conference on Data Mining (SDM)*, 2007.

[15] H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach. *SIGKDD Explorations Newsletter*, 6(1):30–39, 2004.

[16] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pages 179–186, 1997.

[17] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[18] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[19] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, 2005.

[20] D. Tao, X. Tang, X. Li, and X. Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1088–1099, 2006.

[21] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools*. Elsevier, 2005.

[22] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, page 435, 2003.