# RUSBoost: Improving Classification Performance when Training Data is Skewed

**Conference Paper** · December 2008

DOI: 10.1109/ICPR.2008.4761297 · Source: DBLP



**4 authors**, including:

Taghi Khoshgoftaar
Florida Atlantic University
**494** PUBLICATIONS **19,393** CITATIONS
SEE PROFILE

Jason Van Hulse
Florida Atlantic University
**70** PUBLICATIONS **4,129** CITATIONS
SEE PROFILE

# RUSBoost: Improving Classification Performance when Training Data is Skewed

Chris Seiffert (chrisseiffert@gmail.com)
Taghi M. Khoshgoftaar (taghi@cse.fau.edu)
Jason Van Hulse (jvanhulse@gmail.com)
Amri Napolitano (anapoli1@fau.edu)
Florida Atlantic University, Boca Raton, Florida, USA

## Abstract

*Constructing classification models using skewed training data can be a challenging task. We present RUSBoost, a new algorithm for alleviating the problem of class imbalance. RUSBoost combines data sampling and boosting, providing a simple and efficient method for improving classification performance when training data is imbalanced. In addition to performing favorably when compared to SMOTEBoost (another hybrid sampling/boosting algorithm), RUSBoost is computationally less expensive than SMOTEBoost and results in significantly shorter model training times. This combination of simplicity, speed and performance makes RUSBoost an excellent technique for learning from imbalanced data.*

## 1 Introduction

The problem of class imbalance is well known throughout the data mining community. When training data is highly skewed (the various classes among the data are unevenly represented), constructing a useful model can be a challenging endeavor. Traditional data mining techniques tend to favor classifying examples as belonging to the overrepresented (majority) class, while in most real-world application domains it is the underrepresented (minority) class that carries the highest cost of misclassification. For example, a model constructed with the goal of detecting a disease that affects only 1 in 100 people can achieve a correct classification rate of 99% by classifying all individuals as being disease free. Clearly, such a model has no real-world value.

Many techniques have been proposed to alleviate the problem of class imbalance. Two such techniques are data sampling and boosting [9]. Data sampling balances the class distribution in the training data by either adding examples to the minority class (oversampling) or removing examples from the majority class (undersampling). Both undersampling and oversampling have their benefits and drawbacks. The primary drawback of undersampling is the loss of information associated with deleting examples from the training data. It has the benefit, however, of decreasing the time required to train models since the training dataset size is reduced. Oversampling, on the other hand, does not result in the loss of information. However, it can lead to overfitting [5] and increased model training times due to increased training dataset size.

Another technique that can be used to improve classification performance is boosting. While data sampling was designed with the class imbalance problem in mind, boosting is a technique that can improve the performance of any weak classifier (whether or not the data is imbalanced). The most common boosting algorithm is AdaBoost [6]. AdaBoost iteratively builds an ensemble of models. During each iteration, example weights are modified with the goal of correctly classifying examples in the next iteration that were incorrectly classified during the current iteration. Upon completion, all constructed models participate in a weighted vote to classify unlabeled examples.

In this paper, we present a novel hybrid data sampling/boosting algorithm called RUSBoost, which is designed to improve the performance of models trained on skewed data. RUSBoost is a variation of another hybrid data sampling/boosting algorithm called SMOTEBoost [3], which has been proposed in recent research. Both RUSBoost and SMOTEBoost introduce data sampling into the AdaBoost algorithm. SMOTEBoost does so using an oversampling technique called SMOTE [2] which creates new minority class examples by extrapolating between existing examples. RUSBoost applies random undersampling (RUS), a technique which randomly removes examples from the majority class. Our recent research has shown that random undersampling performs very well, often outperforming SMOTE, despite its simplicity [8]. The motivation for proposing this alternative to SMOTEBoost is simple: complexity, training time and performance.

SMOTE is a much more complex and time consuming data sampling technique than random undersampling.

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

Therefore, SMOTEBoost (which uses SMOTE) is more complex and time consuming to perform than RUSBoost. Furthermore SMOTE is an oversampling technique which carries the drawback of increased training time. SMOTEBoost magnifies this drawback since boosting requires that an ensemble of models be trained (many models with increased training times must be constructed). On the other hand, RUSBoost decreases the time required to construct a model, which is a key benefit especially when creating an ensemble of models. The main drawback of random undersampling, loss of information, is greatly overcome by combining it with boosting. While certain information may be absent during the construction of one of the models, it will likely be included in constructing other models within the boosting ensemble. Our results show that the simpler and quicker RUSBoost algorithm performs favorably when compared to SMOTEBoost, oftentimes resulting in significantly better classification performance.

## 2 RUSBoost

Figure 1 presents the RUSBoost algorithm. First (step 1), the weights of each example are initialized to $\frac{1}{m}$, where $m$ is the number of examples in the training dataset. Then, $T$ weak hypotheses are iteratively trained as follows. In step 2a, random undersampling is applied to remove majority class examples until N% of the new (temporary) training dataset, $S'_t$, belongs to the minority class. $S'_t$ will, therefore, have a new weight distribution, $D'_t$. In step 2b, $S'_t$ and $D'_t$ are passed to the base learner, $WeakLearn$, which creates the weak hypothesis, $h_t$ (step 2c). The pseudo-loss, $\epsilon_t$, (based on the original training dataset, S, and weight distribution, $D_t$) is calculated in step 2d. In step 2e, the weight update parameter, $\alpha$, is calculated as $\frac{\epsilon_t}{1-\epsilon_t}$. Next, the weight distribution for the next iteration, $D_{t+1}$, is updated (step 2f) and normalized (step 2g). After T iterations of step 2, the final hypothesis, $H(x)$, is returned as a weighted vote of the weak hypotheses.

SMOTEBoost uses the Synthetic Minority Oversampling Technique [2] (SMOTE) to introduce oversampling to the AdaBoost algorithm. SMOTE adds new minority class examples to a training dataset by finding the $k$ nearest neighbors of a minority class example and extrapolating between that example and its neighbors to create new examples. Random undersampling, on the other hand, simply removes majority class examples (randomly) from the training data. Compared to random undersampling, SMOTE is signficantly more computationally complex and time consuming. Further, because SMOTE is an oversampling technique, it results in a larger training dataset and therefore increased model training time, which can further slow down the training time of the boosting algorithm. Finally, while previous research has shown SMOTE to be an effective data

**Algorithm RUSBoost**
**Given:** Set $S$ of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$
Weak learner, $WeakLearn$
Number of iterations, $T$
Desired percentage of total instances to be represented by the minority class, $N$

1 Initialize $D_1(i) = \frac{1}{m}$ for all $i$.
2 Do for $t = 1, 2, ..., T$
  a Create temporary training dataset $S'_t$ with distribution $D'_t$ using random undersampling
  b Call $WeakLearn$, providing it with examples $S'_t$ and their weights $D'_t$.
  c Get back a hypothesis $h_t : X \times Y \rightarrow [0, 1]$.
  d Calculate the pseudo-loss (for $S$ and $D_t$): $\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y))$.
  e Calculate the weight update parameter: $\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$.
  f Update $D_t$: $D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}$.
  g Normalize $D_{t+1}$: Let $Z_t = \sum_i D_{t+1}(i)$. $D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}$.
3 Output the final hypothesis: $H(x) = \underset{y \in Y}{\text{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}$.

**Figure 1. The RUSBoost algorithm**

sampling technique [8], we recently found random undersampling to perform as well or better than SMOTE. The performance of these algorithms will be compared in Section 4, but clearly RUSBoost has numerous advantages over SMOTEBoost.

RUSBoost is a variant of the SMOTEBoost algorithm, and Figure 1 can be modified to represent SMOTEBoost by changing step 2a to:

a Create temporary training dataset $S'_t$ with distribution $D'_t$ using SMOTE

For details on the AdaBoost.M1 algorithm we refer the reader to Freund and Schapire's work [6]. As proposed by Chawla et al., SMOTEBoost requires that the user specify the number of examples to be added to the training data. We choose to add enough examples to achieve a target distribution. The number of examples added (or removed by random undersampling) is, therefore, determined on a per-dataset basis. We select three target distributions for both SMOTEBoost and RUSBoost: 35, 50 and 65. That is, sampling is performed to create datasets where 35%, 50% or

| Dataset | Size | % min | # attr |
|---|---|---|---|
| SP3 | 3541 | 1.33 | 43 |
| MAMMOGRAPHY (MAM) | 11183 | 2.32 | 7 |
| SOLARFLAREF(SLF) | 1389 | 3.67 | 13 |
| CCCS12 (C12) | 282 | 5.67 | 9 |
| PC1 | 1107 | 6.87 | 16 |
| SATIMAGE (SAT) | 6435 | 9.73 | 37 |
| ECOLI4 (ECO) | 336 | 10.42 | 8 |

**Table 1. Dataset caracteristics**

| Dataset | None | AdaB | SmoteB | RusB |
|---|---|---|---|---|
| C12 | 0.8247 | 0.9226 | **0.9568** | 0.9503 |
| ECO | 0.7765 | 0.9186 | 0.9268 | **0.9342** |
| MAM | 0.7851 | 0.8951 | 0.9113 | **0.9403** |
| PC1 | 0.5834 | 0.8442 | **0.8560** | 0.8508 |
| SAT | 0.7481 | 0.9396 | **0.9539** | 0.9450 |
| SLF | 0.5380 | 0.8359 | 0.8565 | **0.8858** |
| SP3 | 0.5076 | 0.6625 | 0.7096 | **0.7871** |
| Average | 0.6805 | 0.8598 | 0.8816 | **0.8991** |

**Table 2. Boosting Algorithm Performance**

65% of the examples in the post-sampling dataset are minority class examples. In all cases, we report only the parameter that results in the best performance. Determining the optimal class distribution is outside the scope of this work, for the sake of fairness, we compare SMOTEBoost's best performance to RUSBoost's best performance.

# 3 Experiments

## 3.1 Datasets

Table 1 provides information about the seven datasets used in our experiments, including the size of the datasets (Size), the percentage of examples belonging to the minority class (% min) and the number of attributes (including the class attribute) in each dataset (# attr). The datasets represent a wide varity of sizes and levels of imbalance. Also, these datasets are from various application domains. CCCS12, SP3 and PC1 are software quality datasets, some of which are proprietary. The Mammography dataset was provided by Dr. Nitesh Chawla [2]. SatImage, Ecoli4 and SolarFlareF (which were transformed to contain a binary class) are all available through the UCI repository [1].

## 3.2 Performance Measure

Traditional performance measures, such as overall accuracy, are inappropriate for evaluating models trained on skewed data. Therefore, we employ Receiver Operating Characteristic curves [7], or $ROC$ curves, to evaluate the models constructed in this study. $ROC$ curves graph true positive rates on the $y$-axis versus the false positive rates on the $x$-axis. The resulting curve illustrates the trade-off between detection rate and false alarm rate. Traditional performance metrics consider only the default decision threshold. ROC curves illustrate the performance across all decision thresholds. For a single numeric measure, the *area under the ROC curve* ($AUC$) is widely used, providing a general idea of the predictive potential of the classifier.

## 3.3 Design Summary

All experiments were performed using 10-fold cross validation. That is, the datasets were split into ten partitions, nine of which were used to train the models, while the remaining partition was used as test data. We performed 10 independent runs of 10-fold cross validation to eliminate any biasing that could occur as a result of the random partitioning process. Boosting was performed using ten iterations, and experiments with different numbers of boosting iterations did not significantly impact the results. For SMOTEBoost and RUSBoost, which require a user-specified parameter (the level of sampling to perform) we selected three parameters (described in Section 2) and present only the performance of the best parameter for each algorithm. The goal in doing so is a fair performance comparison: the best performance of SMOTEBoost to the best performance of RUSBoost. AdaBoost has no such user-specified parameter. Experiments were also performed without using boosting to serve as a base-line for comparison. We select RIPPER [4] as the base learner, since it is commonly used in imbalanced data research, including the proposal of SMOTEBoost [3].

# 4 Empirical Results

Table 2 shows the performance (measured using AUC) of AdaBoost (AdaB), SMOTEBoost (SmoteB) and RUSBoost (RusB) on all seven of the datasets used in our experiments as well as the results without boosting (None). Also included is the average performance across all seven datasets. For each dataset, the best performing algorithm is indicated by a **bold** value. Table 2 shows that for all datasets, the best performing technique is always either RUSBoost or SMOTEBoost. A $t$-test was performed to compare the performances of RUSBoost and SMOTEBoost. An underlined value in the RUSBoost column indicates that RUSBoost performed significantly better than SMOTEBoost at the $\alpha = 5\%$ significance level. Similarly, an underlined value in the SMOTEBoost column indicates that its performance was significantly better than RUSBoost.

Simple boosting (AdaBoost) is very effective at improving classification performance when learning from imbalanced data. On average, using AdaBoost improves the performance of RIPPER from 0.6805 to 0.8598, an increase of over 25%. Using the PC1 and SolarFlare datasets, the

improvement is the greatest (45% and 55%, respectively). Clearly, boosting has great potential for improving the performance of classifiers built on imbalanced data. While AdaBoost successfully improves classification performance in all of our experiments, both RUSBoost and SMOTEBoost perform better than AdaBoost on every dataset used in this study.

The overall performance of RUSBoost (the average performance across all seven datasets) is significantly better than that of SMOTEBoost. While SMOTEBoost improves the average performance across all seven datasets from 0.6805 (without boosting) to 0.8816, RUSBoost performs even better, achieving an AUC of 0.8991. As indicated by the underlined value, the performance of RUSBoost is significantly better than that of SMOTEBoost at the $\alpha = 5\%$ significance level.

For three of the seven datasets, SMOTEBoost resulted in a higher AUC than RUSBoost. However, only one of those three datasets results in a statistically significant difference in performance. While SMOTEBoost results in a (slightly) better performance for the CCCS12 and PC1 datasets, these differences are not significant. SatImage is the only dataset where SMOTEBoost significantly outperforms RUSBoost (0.9539 vs. 0.9450).

On the other hand, RUSBoost resulted in a higher AUC than SMOTEBoost for four of the seven datasets. In all but one of these datasets, the difference in performance was statistically significant. For the Ecoli dataset, RUSBoost performed slightly better than SMOTEBoost but the difference was not significant. For the Mammography, SolarFlare and SP3 datasets, RUSBoost significantly outperformed SMOTEBoost. The largest difference in performance between RUSBoost and SMOTEBoost occurred using the SP3 dataset where RUSBoost achieved an AUC of 0.7871 compared to SMOTEBoost's 0.7096, an improvement of nearly 11%. In summary, RUSBoost significantly outperformed SMOTEBoost on three datasets, while SMOTEBoost only significantly outperformed RUSBoost on one.

## 5 Conclusion

In this work we present RUSBoost, a new technique for learning from skewed datasets. RUSBoost combines random undersampling with boosting, resulting in improved classification performance when training data is imbalanced. RUSBoost is compared to the SMOTEBoost algorithm, proposed by Chawla et al. [3]. Instead of performing a complex oversampling algorithm (SMOTE), we utilize a simple undersampling technique (random undersampling) that has been shown to result in excellent performance when training models on imbalanced data. The result, RUSBoost, is a simple alternative to SMOTEBoost which is computationally less expensive, results in faster model training times, and provides competitive performance.

Our results show that RUSBoost performs as well or better than SMOTEBoost on six of the seven training datasets used in this study. SMOTEBoost only significantly outperformed RUSBoost on one of the seven datasets. Using three of the datasets, RUSBoost performed significantly better than SMOTEBoost. Overall, RUSBoost significantly outperformed SMOTEBoost. RUSBoost accomplishes this superior performance while boasting reduced computational complexity and training times when compared to SMOTEBoost. We highly recommend using RUSBoost as an alternative to SMOTEBoost when constructing learners using imbalanced datasets.

Future work will continue to investigate the performance of RUSBoost. We will evaluate RUSBoost using additional learners and performance metrics, as well as additional datasets. Further, we will compare the performance of RUSBoost to other boosting algorithms designed to address the class imbalance problem.

## References


[1] A. Asuncion and D. Newman. UCI machine learning repository. *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, 2007. University of California, Irvine, School of Information and Computer Sciences.

[2] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Smote: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, (16):321–357, 2002.

[3] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *In Proceedings of Principles of Knowledge Discovery in Databases*, 2003.

[4] W. W. Cohen. Fast effective rule induction. In *In Proc. 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[5] C. Drummond and R. C. Holte. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Data Sets II, International Conference on Machine Learning*, 2003.

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

[7] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[8] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 935–942, Corvallis, OR, USA, June 2007.

[9] G. M. Weiss. Mining with rarity: A unifying framework. *SIGKDD Explorations*, 6(1):7–19, 2004.