Making Class Bias Useful: A Strategy of Learning from Imbalanced Data

Jie Gu[1], Yuanbing Zhou[2], and Xianqiang Zuo[2]

[1] Software School of Tsinghua University
guj05@mails.tsinghua.edu.cn
[2] State Power Economic Research Institute, China
{zhouyuanbing,zuoxinqiang}@chinasperi.com.cn

Abstract. The performance of many learning methods are usually influenced by the class imbalance problem, where the training data is dominated by the instances belonging to one class. In this paper, we propose a novel method which combines random forest based techniques and sampling methods for effectively learning from imbalanced data. Our method is mainly composed of two phases: data cleaning and classification based on random forest. Firstly, the training data is cleaned through the elimination of dangerous negative instances. The data cleaning process is supervised by a negative biased random forest, where the negative instances have a major proportion of the training data in each of the tree in the forest. Secondly, we develop a variant of random forest in which each tree is biased towards the positive class to classify the data set, where a major vote is provided for prediction. In the experimental test, we compared our method with other existing methods on the real data sets, and the results demonstrate the significative performance improvement of our method in terms of the area under the ROC curve(AUC).

1 Introduction

In recent years, learning from imbalanced data has received increasing interest from the community of machine learning and data mining. Imbalanced data sets exhibit skewed class distributions in which almost all instances are belonging to one or more larger classes and far fewer instances belonging to a smaller, but usually more interesting class. This kind of data can be found in many real-world applications, such as power load data, medical data, sales data, etc.

The balance of training set is an underlying assumption for most learning systems, thus the performance of these systems can be influenced greatly when learn from imbalanced data. More specifically, models trained from imbalanced data sets are biased towards the majority classes and intended to ignore the minority but interested classes. This is the class bias problem, which can lead to a poor performance of the classifier when deal with the class that is not biased. As an example, for a data set where only 1% instances are positive, an accuracy of 99% will be achieved simply by classifying all the instances to be negative. Such is a typical classifier with high accuracy but useless when exploited to classify the positive instances.

Class bias is the major reason of the decline of classification performance on imbalanced data, and thus it is by no means trivial to explore a solution to address the problem. In this paper, we propose a strategy of learning from the imbalanced data set, where class bias is utilized to help improving the classification performance in a proper way. Through analyzing the issue in detail, we can conclude that class bias is not the only reason for the loss of performance, and class overlapping is another problem that could also hinder the performance. In our method, firstly, class bias is exploited to settle class overlapping in the data cleaning process. Secondly, we provide a combination of ensemble learning methods and sampling techniques, where the final prediction is made based on biased classifiers. In the experimental evaluation, we compared our method with other existing methods on the real data sets, and the results demonstrate superior performance of the proposed methods.

The remainder of this paper is organized as follows. In Section 2, we give a brief review of the related work. Our method is described in detail in Section 3. Section 4 gives the experimental evaluation on competitive methods. Finally, Section 5 offers the conclusion remarks.

2 Related Work

A number of solutions to the class-imbalance problem were previously proposed both at the data and algorithmic levels [2]. At the data level, these solutions are based on many different forms of re-sampling techniques including under-sampling the majority and over-sampling the minority to balance the class distribution. At the algorithmic level, the frequently used methods includes cost-sensitive classification [3], recognition-based learning [4], etc.

Random forest [1] is an ensemble of unpruned classification or regression trees, trained from bootstrap samples of the training data, using random feature selection in the tree induction process. The classification is made through a majority vote which takes all the decision of the trees into consideration. Random forest shows important performance improvement over the single tree classifiers and many other machine learning techniques. However, random forest also suffers from the class imbalance when learning from the data set. In this paper, we made significant modification to the basic random forest algorithm to tackle the problem of learning form imbalanced data set.

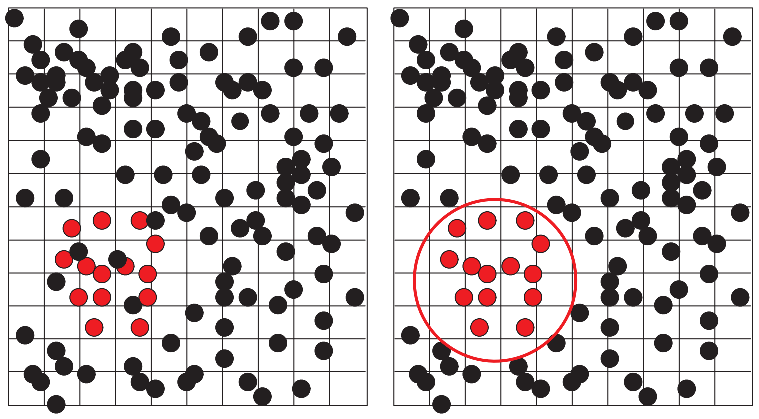
3 Proposed Solutions

The key idea of our method is to utilize the class bias for removing the data, which seriously puzzle classification process, defined as "dangerous". Our method mainly consists of two phases:

1. Data cleaning: The majority of data are preprocessed to eliminate the instances, which may cause degradation of prediction performance.

2. Classification (ranking): We build the model to produce a score for each of the instances in the given prediction data set to indicate their possibility of being positive or negative.

3.1 Data Cleaning



(a) Class overlapping exists before Data Cleaning

(b) Well-defined class boundary emerges after Data Cleaning

Fig. 1. Effect of Data Cleaning

The degradation of performance in many standard classifiers is not only due to the imbalance of class distribution, but also the class overlapping caused by class imbalance. To better understand this problem, imagine the situation illustrated in **Figure 1**. Figure 1(a) represents the original data set without any preprocess. The circle in red and black represent the positive class and the negative class respectively, where an obvious class imbalance exists. Note that in Figure 1(a) there are several negative instances in the region dominated by the positive instances, which presents some degree of class overlapping. These negative instances are considered to be 'dangerous' since it is quite possible for any model trained from this data set to misclassify many positive instances as negative. For cost-sensitive related problems, this issue is even more detrimental. Then a natural requirement is to eliminate the dangerous negative instances, i.e., the black circles in the red region in Figure 1(a). The data set after cleaning, where a well-defined class boundary exists, is represented in Figure 1(b).

There are many existing techniques designed for this data cleaning task, including **Tomek links** [7], **Condensed Nearest Neighbor Rule** [6], etc.

The main defect of these methods lies in their strong dependency on a distance function defined for the data set. However, the most effective form of the distance function can only be expressed in the context of a particular data domain. It is also often a challenging and non-trivial task to find the most effective form of the distance function for a particular data set [8]. Without a well-approximately defined distance function, the result of the data cleaning process is often very poor and cannot eliminate the dangerous negative instances effectively.

The data cleaning method proposed in this paper does not employ the utilization of any distance functions and is more straightforward. In this method, three steps are designed to implement cleaning process as follows:

1. Divide: The data are divided into the minor instances set $\mathcal{P}$ and the major instances set $\mathcal{N}$, and $\mathcal{N}$ is further divided into n subsets $\{\mathcal{N}_1, \mathcal{N}_1, ..., \mathcal{N}_n\}$, where each of them has approximately the same size.
2. Train: For each $\mathcal{N}_i$, we train a random forest RF from the rest instances in $\mathcal{N}$ and the entire $\mathcal{P}$. The trick is that for every classification tree in the forest, the class distribution in the corresponding training data is not balanced, i.e., more negative instances than positive instances.
3. Filter: We remove all instances in $\mathcal{N}_i$, which are are incorrectly classified by RF, from the training data set.

The rationale behind this data cleaning method is quite legible. Most standard learning algorithms assume that maximizing accuracy on a full range of cases is the goal and, therefore, these systems exhibit accurate prediction for the majority class cases, but very poor performance for minority. For such negative-biased classification model, if a negative instance is misclassified as positive, it is reasonable to consider it as dangerous since it must be highly similar with some certain positive instances and thus is responsible for class overlapping. Elimination of these dangerous negative instances from the training data will potentially reduce the false negative rate of the model trained from it. Details of this method is described in Algorithm.1.

3.2 Classification and Ranking

In this subsection, we introduce the method developed to rank the given instances in the testing data set to indicate their possibility of being positive. The basic model is similar with the random forest used in the data cleaning process. The only difference is that the class distribution of the training data for each tree in the forest has been reversed, i.e., more positive instances than negative instances. Several random forests are obtained in this way, and the instances in the prediction data set will be classified by the trained random forests in an iterative process. For each instance, the number of trees in the forest that classifies it as positive are assigned as its score, which represents the possibility of that it is positive. Any instance that receives a score lower than a given threshold will be excluded from the next iteration. At last, all the instances are ranked according to the sum of their scores received in all the iterations.

Algorithm 1 Data Cleaning: Eliminate Dangerous Negative Instances

Input: majority instance set $\mathcal{N}$, minority instance set $\mathcal{P}$, number of subset n, number of trees in the forest l_{tree}, threshold $\epsilon \in (0,1)$

Output:clean data set $D^{'}$ where dangerous negative instances are eliminated from D according to the given parameter ϵ

```
 1: Divide $\mathcal{N}$ randomly into $n$ subsets $\{\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_n\}$, $\forall i, j \in [1, 2, ..., n]$, $|\mathcal{N}_i|$=$|\mathcal{N}_j|$,
    $\mathcal{N}_i \cap \mathcal{N}_j$=$\emptyset$ if $i \neq j$.
 2: $i \leftarrow 0$
 3: repeat
 4:     $i \leftarrow i + 1$
 5:     $\mathcal{N}^{'} = \bigcup_{j=1}^{n} \mathcal{N}_j - \mathcal{N}_i$
 6:     for $m = 1$ to $l_{tree}$ do
 7:         Randomly sample a subset $\mathcal{P}_{sub}$ from $\mathcal{P}$ and a subset $\mathcal{N}_{sub}$ from $\mathcal{N}^{'}$, where
            $|\mathcal{P}_{sub}| \ll |\mathcal{N}_{sub}|$
 8:         Learning a classification tree $T_m$ form $\mathcal{N}_{sub} \cup \mathcal{P}_{sub}$
 9:     end for
10:     for each instance $X_n \in \mathcal{N}_i$ do
11:         if more than $\epsilon \times l_{tree}$ tree classifiers classify $X_n$ as positive then
12:             Eliminate $X_n$ from $\mathcal{N}_i$
13:         end if
14:     end for
15: until $i$=$n$
16: $\mathcal{N}_{out}$=$\bigcup_{j=1}^{n} \mathcal{N}_j$
17: Return $\mathcal{N}_{out}$ as the output
```

Algorithm 2 Classification and Ranking

Input: cleaned training data set $\mathcal{D}_t$, prediction data set $\mathcal{D}_p$, number of iterations n, number of trees in each random forest l_{tree}, threshold $\epsilon \in (0, 1)$

Output: $\mathcal{D}_t$ with each instance being assigned with a score

```
 1: Divide $\mathcal{D}_t$ into positive subset $\mathcal{D}_{tp}$ and negative subset $\mathcal{D}_{tn}$
 2: initiate array $\mathcal{SCORE}[\ ]$ of length $|\mathcal{D}_p|$
 3: $i \leftarrow 0$
 4: **repeat**
 5:    $i \leftarrow i + 1$
 6:    **for** $m = 1$ to $l_{tree}$ **do**
 7:       Randomly sample a subset $\mathcal{P}_{sub}$ from $\mathcal{D}_{tp}$ and a subset $\mathcal{N}_{sub}$ from $\mathcal{D}_{tn}$, where
          $|\mathcal{P}_{sub}| \gg |\mathcal{N}_{sub}|$
 8:       Learning a classification tree $T_m$ form $\mathcal{N}_{sub} \cup \mathcal{P}_{sub}$
 9:    **end for**
10:    **for** each instance $X_n \in \mathcal{D}_p$ **do**
11:       **for** each classification tree $T_m$ in the random forest **do**
12:          **if** $T_m$ classify $X_n$ as positive **then**
13:             $\mathcal{SCORE}[n] \leftarrow \mathcal{SCORE}[n] + 1$
14:          **end if**
15:       **end for**
16:       **if** less than $\epsilon \times l_{tree}$ tree classifiers classify $X_n$ as positive **then**
17:          Eliminate $X_n$ from future iterations
18:       **end if**
19:    **end for**
20: **until** $i$=$n$
21: Return $\mathcal{SCORE}$ as the output
```

The idea behind this method is that again we exploit the bias of the classifiers trained from imbalanced data set as in the data cleaning process. Since the positive instances dominate the training data set, each tree is biased towards correctly classifying the positive instance. Then there is a strong possibility for a positive instance to receive a higher score and a negative instance to receive a lower one. By excluding the instances that received a score lower than a specified threshold, we restrict the model to focus on the hard to classify instances.

4 Experimental Evaluation

In this section , we run a series of experiments to evaluate the classification performance of the proposed methods. The decision tree in the random forest is implemented using J48 in the Weka machine learning tool[3].

4.1 Data Set

8 UCI data sets which have different degrees of imbalance were used in our data set. Information of these data are listed in Table 1, including data size, number of attributes, the target attributes, and class distributions. For data sets with more than 1 classes, we choose the class with least instances as the minority class and consider the remainder as the majority class. More details about these data sets can be found in the UCI data page[4].

Table 1. Information about the data set

Data Set	Size	Attributes	Class	Class Distribution	ClassNumber
balance	625	4	Balance	1:12.0	3
flags	194	28	White	1:10.4	7
haberman	306	3	Die	1:2.81	2
letter	20000	16	A	1:24.3	26
nursery	12960	8	not_recom	1:38.2	5
pima	768	80	1	1:2.01	2
sat	6435	36	6	1:3.23	3
vehicle	846	18	opel	1:3.31	4

4.2 AUC Score

The performance of each method is measured by means of ROC [9], which represents the false positive rate on the horizontal axis of a graph and the true positive

[3] http://www.cs.waikato.ac.nz/ml/weka/

[4] http://www.ics.uci.edu/mlearn /MLRepository.html

rate on the vertical axis. A curve is produced by varying the threshold on a classification model's numeric output. The Area Under the Curve is a widely used performance measurement of the classification accuracy on imbalanced data set.

4.3 Comparison with Existing Methods

The proposed methods are compared with 5 other popular techniques used for imbalanced learning, including SMOTE, under-sampling(Under), over-sampling(Over), Tomek links(Tomek), Condensed Nearest Neighbor Rule(CNN). RF refers to our proposed random forest based methods. For each data set, 10 times 10-fold cross validation are executed. In order to reduce the bias introduced by the sampling process, within each fold the learning algorithm is repeated for 20 times. Finally, the averaged AUC score is reported in Figure 2.

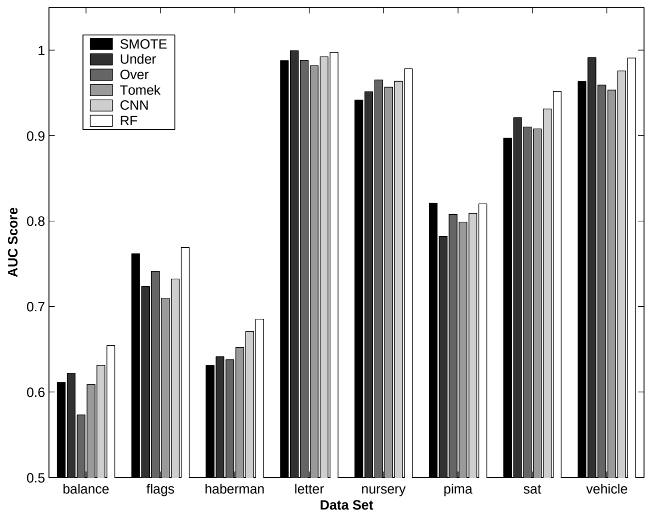


Fig. 2. AUC Scores on 8 UCI Data Sets

RF achieves the best performance on 5 of the data sets. For the other 3 data sets, it is almost as good as the method with the highest AUC score. The similarity function based methods, such as SMOTE and Tomek, does not outperform the simple over-sampling and under-sampling approach since their performance depends greatly on the similarity function and it is not an easy task to find a suitable similarity function for a given data set. While our proposed method does not rely on the similarity function, it outperforms its competitors in terms of prediction performance.

4.4 Influence of Class Distribution

In the classification step, we are reversing the class distribution of the training data for each tree in the forest. In this subsection, we test the influence of the class distribution on the classification accuracy of the proposed method(RF). For each data set, we changed the distribution of the positive instances and the negative instances and tested the performance of the trained model. From Figure 3 we can see that the AUC scores can be largely influenced by the distribution. Our method can achieve the best performance under a certain proper class distribution. As the ration of $Minor/Major$ decreases, the AUC score also decreases.

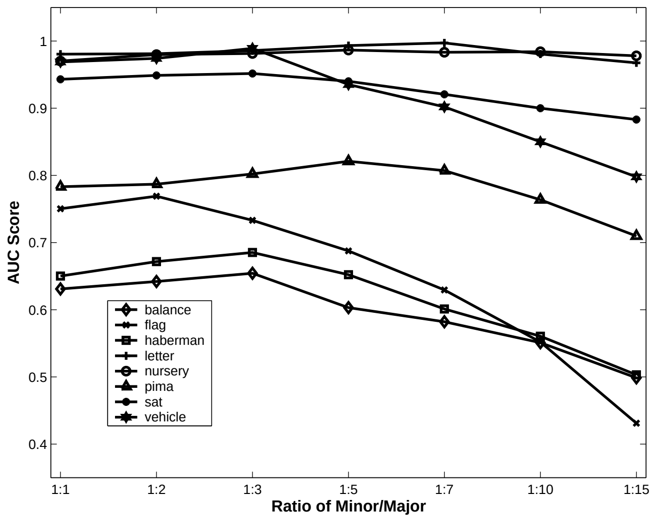


Fig. 3. Influence of Class Distribution

5 Summary

In this paper , we propose a strategy of learning from imbalanced data set. The main idea of our method is to make the class bias useful when deal with imbalanced data set. In the data cleaning process, the class bias is used to eliminate dangerous negative instances and further address the class overlapping. In the classification process, each classifier is trained to be biased towards the positive class, that is, the class distribution has been reversed. The testing set is classified in an iterative way, where the order in which the instances being labeled reflects their possibility of being positive.

References

1. Breiman.L. *Random Forest.* In *Machine Learning(2001), 45,5-32.*
2. Nitesh V.Chawla, Nathalie Japkowicz, and Aleksander Kolcz. *Editorial: Special Issue on Learning from Imbalanced Data Sets.* In *Sigkdd Explorations(2004), Volume 6, Issue 1, Page 1.*
3. K. M. Ting. *A comparative study of cost-sensitive boosting algorithms.* In *Proceedings of Seventeenth In- ternational Conference on Machine Learning, pages 983-990, Stanford, CA, 2000.*
4. P. Juszczak and R. P. W. Duin. *Uncertainty sampling methods for one-class classifiers.* In *Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets, 2003.*
5. Nitesh V.Chawla, Kevin W.Bowyer, Lawrence O.Hall, W.Philip Kegelmeyer. *SMOTE: Synthetic Minority Over-sampling TEchnique.* In *Journal of Artificial Intelligence Research(2002), 321-357.*
6. Hart, P.E. *The Condensed Nearest Neighbor Rule.* In *IEEE Transactions on Information Theory IT-14 (1968), 515C516.*
7. Tomek, I. *Two Modifications of CNN.* In *IEEE Transactions on Systems Man and Communications SMC-6 (1976), 769C772.*
8. Charu C.Aggarwal. *Towards Systematic Design of Distance Functions for Data Mining Applications.* In *Proceedings of the International Conference on Knowledge Discovery and Data Mining(2003).*
9. A.P.Bradley. *The use of the area under the ROC curve in the evaluation of machine learning algorithms.* In *Pattern Recognition, 30(7):1145-1159, 1997.*
10. A.P.Bradley. *Cost-sensitive Learning by Cost-proportionate Example Weighting.* In *Proceedings of the 3rd IEEE International Conference on Data Mining, pages 435C442, Melbourne, FL, 2003..*