Comparing Boosting and Cost-Sensitive Boosting With Imbalanced Data

Fangjun Wu
School of Information Technology
Jiangxi University of Finance and Economics
NanChang, 330013, China
wufangjun@jxufe.edu.cn

Abstract

Class imbalance problem has emerged as one of the crucial issues in machine learning and data mining communities since there is increasing growth and availability of real world data distributed skew or unequal misclassification costs of the minority and majority classes. This paper compares the performance of several boosting and cost-sensitive boosting methods in terms of their capabilities in dealing with the class imbalance problem by using evaluation metrics, precision, F-Measure, Geometric mean (G-mean), and the area under receiver operating characteristics curve (AUC) on five NASA benchmark imbalanced datasets (JM1, KC1, KC2, PC1 and CM1). The learning algorithms studied in this paper include Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, and cost-sensitive classifiers based on the former two respectively. The experimental results show that it is difficult to say which one is the best for handling class imbalance without consideration of evaluation metrics.

Keywords: *Classification, Class Imbalance, AdaBoost, Cost-Sensitive Learning*

1. Introduction

With the increasing growth and availability of real world data distributed skew or unequal misclassification costs of the minority and majority classes, class imbalance problem has emerged as one of the crucial issues in machine learning and data mining communities [1-7]. Although the number of samples from minority classes is much smaller than from majority classes, classifiers have to identify a minor but important case, such as software fault-proneness analysis [1], credit card fraud detection, rare medical diseases diagnosis, and oil spill detection in satellite radar images.

Many methods have been developed to address this problem [8-19]. Among them, data sampling is a popular method, which contains majority undersampling and minority oversampling. The former creates a subset of the original data set by deleting instances from the majority class; conversely, the latter creates a superset of the original data set by copying or creating new instances into the minority class. Synthetic sampling with data generation (SMOTE) [9], adaptive synthetic sampling (Borderline-SMOTE) [10], RUSBoost [13], Ramoboost [14], IRUS [16], EDAOS [18] and SBoost [19] are famous data sampling approaches and have achieved satisfactory results.

In addition to data sampling, cost-sensitive extensions of existing algorithms are other kinds of methods proposed for handling class imbalance by adding costs, such as AdaCost [8], AdaC1 & AdaC2 & AdaC3 [11] and cost-sensitive variations of AdaBoost & RealBoost & LogitBoost [15].

Each method has its benefits and drawbacks. Some authors support boosting as being better than data sampling [13]. Others find there is no significant difference between bagging and boosting [4]. Still others discover that cost-sensitive learning outperforms data sampling [17], and oversampling performs better than undersampling [17]. However, there is no study focusing on comparing boosting and cost-sensitive learning. This paper will address this problem.

The remainder of this paper is organized as follows. Section 2 provides an overview of AdaBoost and cost-sensitive AdaBoost. The components of our experimental design are discussed in Section 3. The experimental results are analyzed in Section 4. Conclusions are presented in Section 5.

2. An overview of AdaBoost and Cost-Sensitive AdaBoost

AdaBoost is one of the most popular boosting algorithms, proposed by Yoav Freund and Robert Schapire [20]. It iteratively increases weights of misclassified instances, and decreases weights of correctly classified instances, thus constructing a "strong" classifier H:X→Y as a linear combination of "simple" "weak" classifiers on the condition that performances of "weak" classifiers are better than guesses (error rate is less than 0.5 for a binary classification problem), where X is the instance space, and contains n-tuple of attribute values; Y={-1, +1} is label set, where y_i returns the label for the instance x_i. Its pseudocode is described in Algorithm 1.

Algorithm 1 Pseudocode for AdaBoost [20]
Input: the training data set T, the test data set S, a "weak" classifiers h, and the maximum number of iterations N.
Initialize the initial weight vector, that $D^1 = (\frac{1}{m}, \dots, \frac{1}{m})$, where m is the size of T. We allow users to specify the initial vector.
For t=1,…,N
1. Train base learner $h_t: X \rightarrow Y$ using distribution $\mathbf{w}^t$ over T and the unlabeled data set S;
2. Set $\beta_t = \frac{1}{2}\log(\frac{\sum_{i,y_i=h_t(x_i)} D_i^t}{\sum_{i,y_i \neq h_t(x_i)} D_i^t})$, where $\sum_{i,y_i=h_t(x_i)} D_i^t > \sum_{i,y_i\neq h_t(x_i)} D_i^t$; (1)
3. Update the new weight vector: $D_i^{t+1} = \frac{D_i^t \exp(-\beta_t h_t(x_i) y_i)}{Z_t} = \begin{cases} \frac{D_i^t \exp(-\beta_t)}{Z_t}, & h_t(x_i) = y_i \\ \frac{D_i^t \exp(\beta_t)}{Z_t}, & h_t(x_i) \neq y_i \end{cases}$ (2)

   Where Z_t is a normalization factor.

Output the final classifier $H(x) = \mathrm{sign}\left(\sum_{t=1}^{N} \beta_t h_t(x)\right) = \mathrm{argmax}(\sum_{t=1}^{N} \beta_t h_t(x))$

Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong and Yang Wang extended AdaBoost for cost-sensitive learning by adding misclassification cost into the weight update strategy, and proposed three cost-sensitive boosting algorithms, namely AdaC1, AdaC2 and AdaC3 [11]. Formally, the problem that AdaC1, AdaC2 and AdaC3 solved is as followed. Given a number of labeled training data T, some unlabeled test data S and misclassification costs of the minority and majority classes, namely C_P and C_N, the objective is to train a "strong" classifier H: X→Y that reduce misclassification cost on the unlabeled data set S to a large extent.

AdaC1, AdaC2 and AdaC3 keep the same framework of AdaBoost, but update weights differently, namely adding misclassification costs of the minority and majority classes to the different places relative to the bracket of Eq.(2), either at the front of, or in, or outside of the bracket.

Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong and Yang Wang pointed out [11] that AdaC2 and AdaC3 are sensitive to the cost setups. Furthermore, they conclude that AdaC2 is superior to its rivals.

3. Experiments

In this section, we set up experiments to investigate Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, and cost-sensitive classifiers based on the former two respectively, also known as CSCLogistic and CSC AdaBoost, in terms of their capabilities in dealing with the class imbalance problem.

3.1 Data sets

In this paper, five NASA projects (JM1, KC1, KC2, PC1 and CM1) are selected as the class imbalance problems inherent in the data which hinder the learning from building an effective classification model to distinguish fault-prone modules from the non-fault-prone modules. They can be downloaded from NASA metrics data program repository (http://promise.site.uottawa.ca/SERepository) [21]. They have 22 attributes, where 21 are metric attributes, and 1 is fault-proneness attribute. The former 21 metric attributes are more popular, such as lines of code (shortly LOC), McCabe cyclomatic complexity metric [22], Halstead metrics, Chidamber-Kemerer metrics suite [23]. The last attribute is label: label +1 denotes fault-proneness which is treated as the positive class and label -1 represents the non-fault-proneness which is treated as the negative class.

3.2 Weak Classifier

This paper uses the well-known Logistic regression as the weak classifier based on two reasons: (1) it is well-known and widely used; (2) it is a standard statistical technique for classification. Moreover, the default parameters for it are set as recommended in WEKA [24]. In most cases, the impact of varying the parameters is small. The default parameters of the Logistic regression include the following [24]: maxIts -1, ridge 1.0E-8.

3.3 Evaluation Criteria

A large number of evaluation criteria have been used to evaluate and compare classification models [25, 26]. Almost all evaluation criteria are represented on the basis of the confusion matrix, such as accuracy, precision, recall, TPR, FPR, F-measure, G-Mean, GMPR, and AUC. Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong and Yang Wang stated that the learning objective of class imbalance problem is either to achieve high recognition success of the positive class or to balance recognition ability between positive class and negative class [11]. As recommended in Ref. [25], the classification performances are evaluated by precision, F-measure, G-Mean, and AUC in this paper.

4. Empirical results

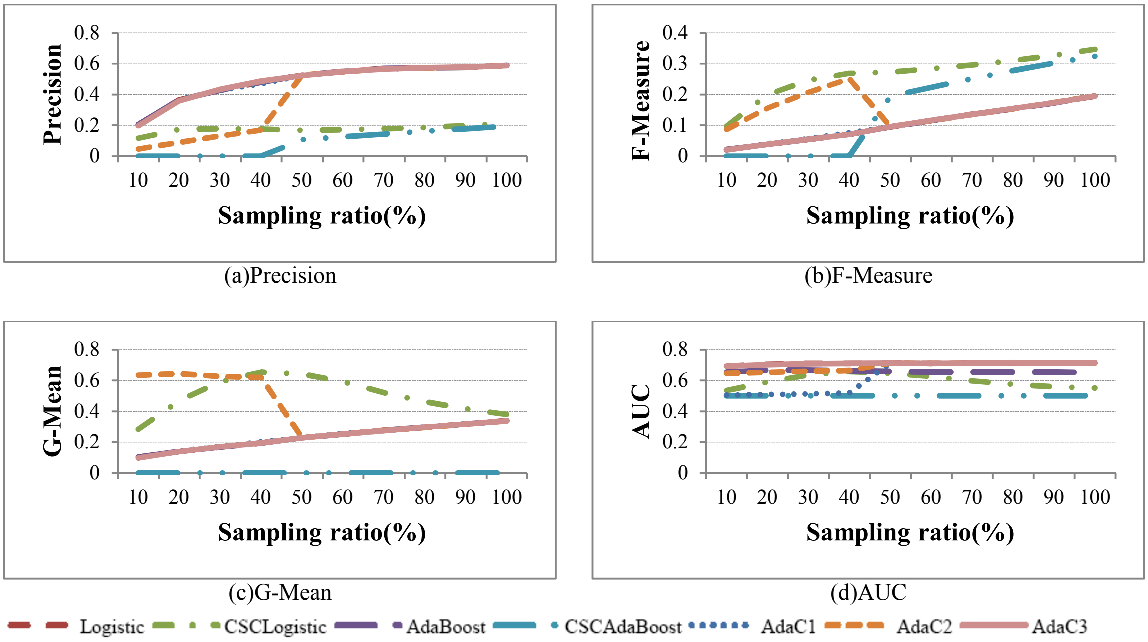

Figure 1. Performance of Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, CSCLogistic and CSC AdaBoost across various class imbalanced ratios on the data set JM1, where C_P=1 and C_N=0.1.

The performances in term of evaluation criteria are the average results of 50 repeats by random. All performances are always generated by tenfold cross-validation of classification to avoid sampling bias. The number of iterations N is set to 10.

Our experiment is divided into three parts. The first is to observe by how many degrees different class imbalanced ratios affect classification performances. The second is to investigate by how many degrees different data sets affect classification performances. The last is to survey by how many degrees different cost setups of the minority and the majority classes, namely C_P and C_N, affect classification performances.

4.1 Various Class Imbalanced Ratios

In order to observe by how many degrees different class imbalanced ratios affect classification performances, the minority class of data set JM1 is sampled from 10% to 100%, at the same time as keeping its majority class the same. Moreover, C_P=1 and C_N=0.1. The result is shown in Figure 1.

From Figure 1, some general conclusions can be drawn as follows:

(1) CSC AdaBoost always achieves the lowest precision, G-Mean, and AUC values regardless how many sampling ratios of the minority class of data set JM1.

(2) AdaC2 varies precision, F-Measure, and G-Mean values largely at different sampling ratio of the minority class of data set JM1.

(3) AdaC3 always achieves the highest precision and AUC values regardless how many sampling ratios of the minority class of data set JM1.

(4) AdaC3 and Logistic regression nearly achieve the same Precision, F-Measure, G-Mean, and AUC values on the data set JM1.

(5) With the decreasing sampling ratio of the minority class, class distribution is more skewed. Among three cost-sensitive AdaBoost algorithms, AdaC1 is the worst at dealing with the class imbalance problem, AdaC2 is modern, and AdaC3 is the best in term of AUC.

4.2 Different Data Sets

Table 1. Precision values of Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, CSCLogistic and CSC AdaBoost across various data sets JM1, KC1, KC2, PC1 and CM1, where C_P=1 and C_N=0.1, C_N=0.2, C_N=0.3.

Costs	Datasets	Logistic	CSCLogistic	AdaBoost	CSCAdaBoost	AdaC1	AdaC2	AdaC3
C_N=0.1	JM1	0.5886	0.2118	0.5886	0.1935	0.5873	0.5884	0.5865
	CM1	0.3390	0.2331	0.3390	0.0984	0.3052	0.2189	0.3433
	KC1	0.6097	0.2730	0.6097	0.1546	0.6096	0.6055	0.6089
	KC2	0.6146	0.3679	0.6145	0.2050	0.6160	0.6168	0.6192
	PC1	0.3301	0.2089	0.3301	0	0.3473	0.1940	0.3263
C_N=0.2	JM1	0.5887	0.2943	0.5887	0.1935	0.5886	0.5869	0.5887
	CM1	0.3529	0.3052	0.3529	0	0.3402	0.2151	0.3523
	KC1	0.6068	0.3375	0.6068	0	0.6001	0.2948	0.6046
	KC2	0.6163	0.4438	0.6164	0.2050	0.6012	0.4106	0.6153
	PC1	0.3344	0.2335	0.3344	0	0.3234	0.1989	0.3121
C_N=0.3	JM1	0.5886	0.3926	0.5886	0	0.5874	0.3197	0.5877
	CM1	0.3388	0.3154	0.3388	0	0.3373	0.2164	0.1619
	KC1	0.6078	0.4017	0.6078	0	0.6026	0.2992	0.6062
	KC2	0.6143	0.5078	0.6143	0	0.6176	0.4126	0.6150
	PC1	0.3407	0.2895	0.3407	0	0.3327	0.1906	0.1302

In order to investigate by how many degrees different data sets affect classification performances, five NASA projects (JM1, KC1, KC2, PC1 and CM1) are selected as data sets. Their imbalance ratio, namely the ratios between fault-prone and non-fault-prone modules vary largely, from 6.9432% to 20.4981%. That is to say, JM1 contains 10885 modules, in which

2106 are fault-prone; KC1 contains 2109 modules, in which 326 are fault-prone; KC2 contains 522 modules, in which 107 are fault-prone; CM1 contains 498 modules, in which 49 are fault-prone; PC1 contains 1109 modules, in which 77 are fault-prone. We set C_P=1 and change C_N from 0.1 to 0.3. The results are shown in Tables 1~4.

Table 2. F-Measure values of Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, CSCLogistic and CSC AdaBoost across various data sets JM1, KC1, KC2, PC1 and CM1, where C_P=1 and C_N=0.1, C_N=0.2, C_N=0.3.

Costs	Datasets	Logistic	CSCLogistic	AdaBoost	CSCAdaBoost	AdaC1	AdaC2	AdaC3
C_N=0.1	JM1	0.1953	0.3463	0.1953	0.3242	0.1949	0.1949	0.1945
	CM1	0.1907	0.3484	0.1907	0.1792	0.2738	0.3306	0.2011
	KC1	0.3165	0.4151	0.3165	0.2678	0.3175	0.3162	0.3172
	KC2	0.4913	0.5175	0.4911	0.3402	0.4922	0.4916	0.4944
	PC1	0.1385	0.2923	0.1385	0.0000	0.1640	0.3091	0.1362
C_N=0.2	JM1	0.1948	0.4150	0.1948	0.3242	0.1948	0.1945	0.1951
	CM1	0.2079	0.3868	0.2079	0.0000	0.2042	0.3251	0.2017
	KC1	0.3162	0.4528	0.3162	0.0000	0.3203	0.4268	0.3148
	KC2	0.4909	0.5648	0.4908	0.3402	0.4976	0.5499	0.4919
	PC1	0.1431	0.2473	0.1431	0	0.1410	0.3144	0.1317
C_N=0.3	JM1	0.1956	0.4269	0.1956	0	0.1953	0.4271	0.1949
	CM1	0.1982	0.3399	0.1982	0	0.1988	0.3261	0.2662
	KC1	0.3153	0.4672	0.3153	0	0.3168	0.4314	0.3170
	KC2	0.4907	0.5848	0.4907	0	0.4979	0.5524	0.4904
	PC1	0.1433	0.2425	0.1433	0	0.1436	0.3001	0.2282

Table 3. G-Mean values of Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, CSCLogistic and CSC AdaBoost across various data sets JM1, KC1, KC2, PC1 and CM1, where C_P=1 and C_N=0.1, C_N=0.2, C_N=0.3.

Costs	Datasets	Logistic	CSCLogistic	AdaBoost	CSCAdaBoost	AdaC1	AdaC2	AdaC3
C_N=0.1	JM1	0.3388	0.3807	0.3388	0	0.3384	0.3383	0.3381
	CM1	0.3582	0.7202	0.3582	0	0.4820	0.7052	0.3708
	KC1	0.4565	0.7076	0.4565	0	0.4574	0.4567	0.4572
	KC2	0.6181	0.7315	0.6180	0	0.6187	0.6179	0.6202
	PC1	0.2933	0.6477	0.2933	0	0.3244	0.7623	0.2908
C_N=0.2	JM1	0.3383	0.6472	0.3383	0	0.3383	0.3380	0.3385
	CM1	0.3772	0.6772	0.3772	0	0.3751	0.6993	0.3699
	KC1	0.4565	0.7197	0.4565	0	0.4611	0.7153	0.4554
	KC2	0.6174	0.7627	0.6172	0	0.6274	0.7588	0.6186
	PC1	0.2988	0.4958	0.2988	0	0.2969	0.7618	0.2859
C_N=0.3	JM1	0.3391	0.6218	0.3391	0	0.3388	0.6572	0.3385
	CM1	0.3678	0.5797	0.3678	0	0.3688	0.6988	0.6546
	KC1	0.4556	0.6880	0.4556	0	0.4575	0.7190	0.4573
	KC2	0.6177	0.7554	0.6177	0	0.6239	0.7611	0.6172
	PC1	0.2985	0.4478	0.2985	0	0.2995	0.7401	0.7058

Table 4. AUC values of Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, CSCLogistic and CSC AdaBoost across various data sets JM1, KC1, KC2, PC1 and CM1, where C_P=1 and C_N=0.1, C_N=0.2, C_N=0.3.

Costs	Datasets	Logistic	CSCLogistic	AdaBoost	CSCAdaBoost	AdaC1	AdaC2	AdaC3
C_N=0.1	JM1	0.7136	0.5508	0.6519	0.5	0.7135	0.7135	0.7136
	CM1	0.7819	0.7210	0.6903	0.5	0.5934	0.7060	0.7858
	KC1	0.7993	0.7221	0.7216	0.5	0.8000	0.7988	0.7989
	KC2	0.8163	0.7428	0.7717	0.5	0.8160	0.8151	0.8177
	PC1	0.8149	0.6746	0.7247	0.5	0.5462	0.7624	0.8142
C_N=0.2	JM1	0.7136	0.6494	0.6515	0.5	0.7136	0.7136	0.7136
	CM1	0.7836	0.6986	0.6984	0.5	0.5576	0.7003	0.7875

	KC1	0.7991	0.7204	0.7209	0.5	0.5959	0.7174	0.7990
	KC2	0.8149	0.7629	0.7707	0.5	0.6760	0.7621	0.8160
	PC1	0.8149	0.5993	0.7259	0.5	0.5382	0.7620	0.8122
C_N=0.3	JM1	0.7136	0.6471	0.6518	0.5	0.5487	0.6577	0.7137
	CM1	0.7829	0.6408	0.6996	0.5	0.5556	0.7000	0.6656
	KC1	0.7994	0.7031	0.7212	0.5	0.5945	0.7209	0.7997
	KC2	0.8159	0.7586	0.7713	0.5	0.6753	0.7644	0.8152
	PC1	0.8150	0.5853	0.7275	0.5	0.5390	0.7411	0.7314

From Tables 1~4, it can be seen that all classifiers achieve the highest precision, F-Measure, G-Mean, and AUC values on data set KC2 among five NASA projects (JM1, KC1, KC2, PC1 and CM1). However, both precision values of the positive class on PC1 and CM1 are less than 0.5.

4.2 Cost Setups

In order to observe by how many degrees different cost setups of the minority and majority classes, namely C_P and C_N, affect classification performances, we fix C_P=1, meantime change C_N from 0.1 to 0.9 on data set JM1. The result is shown in Figure 2.

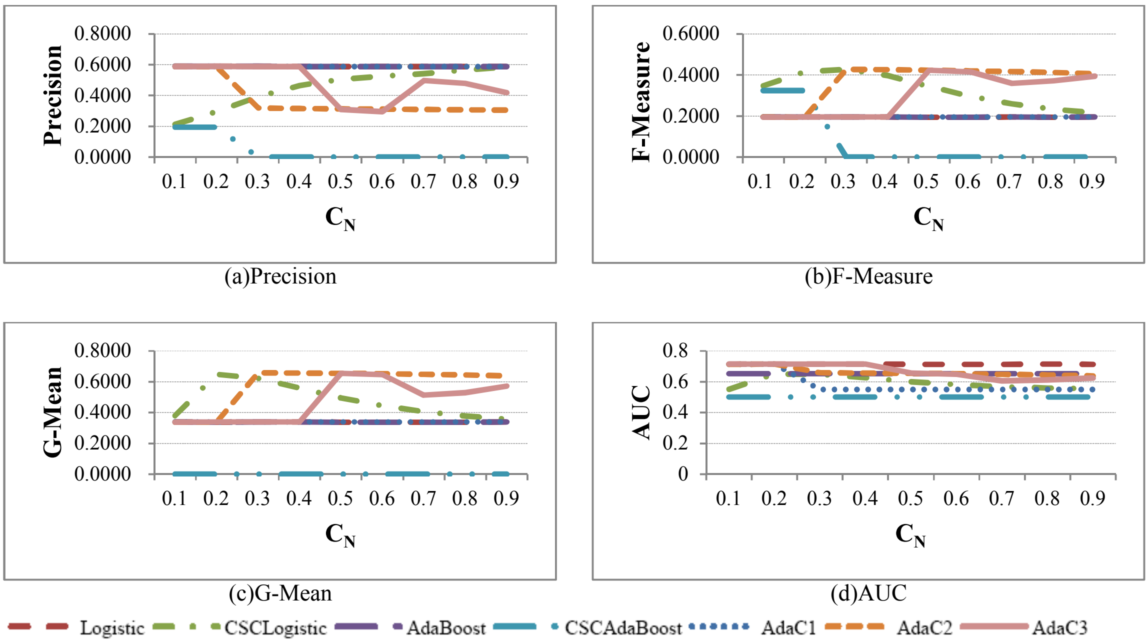

Figure 2. Performance of Logistic regression, AdaBoost, AdaC1, AdaC2, AdaC3, CSCLogistic and CSC AdaBoost across various cost setups on the data set JM1.

From Figure 2, the following statements can be made:

(1) AdaC2 and AdaC3 vary precision, F-Measure, and G-Mean values greatly at different C_N. By contrast, AdaC1 maintains almost the same precision, F-Measure, and G-Mean values regardless the values of C_N. For example, when C_P=1 and C_N=0.2, AdaC1 achieves its highest precision value 0.5886; when C_P=1 and C_N=0.1, it achieves its lowest precision value 0.5865. When C_P=1 and C_N=0.2, AdaC2 achieves its highest precision value 0.5884; when C_P=1 and C_N=0.9, it achieves its lowest precision value 0.3046. When C_P=1 and C_N=0.2, AdaC3 achieves its highest precision value 0.5887; when C_P=1 and C_N=0.6, it achieves its lowest precision value 0.2949. In this sense, AdaC2 and AdaC3 are more sensitive to the cost setups, which are consistent with Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong and Yang Wang's viewpoint [11].

(2) AdaC1, AdaC2, and AdaC3 all vary AUC values greatly at different C_N, from 0.5486 to 0.7136, 0.6380 to 0.7136, 0.6119 to 0.7137 respectively. For example, when C_P=1 and C_N=0.2, AdaC1 achieves its highest AUC value 0.7136; when C_P=1 and C_N=0.5, it achieves its lowest AUC value 0.5485. When C_P=1 and C_N=0.2, AdaC2 achieves its highest AUC value 0.7136; when C_P=1 and C_N=0.9, it achieves its lowest AUC value 0.6380. When C_P=1 and C_N=0.3, AdaC3 achieves its highest AUC value 0.7137; when C_P=1 and C_N=0.7, it achieves its lowest AUC value 0.6064. In this sense, AdaC1, AdaC2 and AdaC3 are all sensitive to the cost setups.

5. Summary

This paper has presented the results of a comprehensive suite of experiments comparing the performance of several boosting and cost-sensitive boosting methods in the context of learning from imbalanced data on five NASA benchmark imbalanced datasets (JM1, KC1, KC2, PC1 and CM1). The experiments, in which three parts were trained and evaluated, varied key factors including class imbalanced distribution, data sets and cost setups of the minority and majority classes, namely C_P and C_N. All models were evaluated using precision, F-Measure, G-mean, and AUC, providing a complete perspective on classification performance. The experimental results demonstrate that it is difficult to say which one is the best among AdaC1, AdaC2, and AdaC3 in terms of their capabilities in dealing with the class imbalance problem.

6. Acknowledgement

This work has been partially supported by the Natural Science Foundation of China (Project No. 61163007) and Natural Science Foundation of Jiangxi (Project No. 20114BAB211019).

7. References

[1] Beiyang Wang and Lifeng Wang, "Analysis of defects propagation in software system based on weighted software networks", Journal of Convergence Information Technology, vol.7, no.17, pp.63-77, 2012.
[2] Haibo He and Edwardo A. Garcia, "Learning from imbalanced data", IEEE Transactions on Knowledge and Data Engineering, vol.21, no.9, pp.1263-1284, 2009.
[3] Guohua Liang and Chengqi Zhang, "An empirical evaluation of bagging with different algorithms on imbalanced data", In Proceedings of the ADMA 2011, Part I, LNAI 7120, pp.339-352, 2011.
[4] Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano, "Comparing boosting and bagging techniques with noisy and imbalanced data", IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, vol.41, no.3, pp.552-568, 2011.
[5] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches", IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, vol.42, no. 4, pp.463-484, 2012.
[6] Camelia Lemnaru and Rodica Potolea, "Imbalanced classification problems: systematic study, issues and best practices", Lecture Notes in Business Information Processing, vol.102, pp.35-50, 2012.
[7] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Andres Folleco, "An empirical study of the classification performance of learners on imbalanced and noisy software quality data", Information Sciences, in press.
[8] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan, "AdaCost: misclassification cost-sensitive boosting", In Proceedings of International Conference on Machine Learning, pp.97-105, 1999.
[9] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer, "SMOTE: synthetic minority over-sampling technique", Journal of Artificial Intelligence Research, vol.16, pp.321-357, 2002.

[10] Hui Han, Wenyuan Wang, and Binghuan Mao, “Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning”, Lecture Notes in Computer Science, vol.3644, pp.878-887, 2005.
[11] Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, and Yang Wang, “Cost-sensitive boosting for classification of imbalanced data”, Pattern Recognition, vol.40, no.12, pp.3358-3378, 2007.
[12] Dennis J. Drown, Taghi M. Khoshgoftaar, and Naeem Seliya, “Evolutionary sampling and software quality modeling of high-assurance systems”, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, vol.39, no.5, pp.1097-1107, 2009.
[13] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano, “RUSBoost: a hybrid approach to alleviating class imbalance”, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, vol.40, no.1, pp.185-197, 2010.
[14] Sheng Chen, Haibo He, and Edwardo A. Garcia, “Ramoboost: ranked minority oversampling in boosting”, IEEE Transactions on Neural Networks, vol.21, no.10, pp.1624-1642, 2010.
[15] Hamed Masnadi-Shirazi, and Nuno Vasconcelos, “Cost-sensitive boosting”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.33, no.2, pp.294-309, 2011.
[16] Muhammad AtifTahir, Josef Kittler, and Fei Yan, “Inverse random under sampling for class imbalance problem and its application to multi-label classification”, Pattern Recognition, vol.45, pp.3738-3750, 2012.
[17] Kate McCarthy, Bibi Zabar, and Gary Weiss, “Does cost-sensitive learning beat sampling for classifying rare classes?”, In Proceedings of UBDM ’05, Chicago, Illinois, USA, pp.69-75, August 21, 2005.
[18] Liu Wei, Zhang Dongmei, Li Yang, "A novel over-sampling method based on EDAs for learning from imbalanced data sets", JCIT: Journal of Convergence Information Technology, vol. 6, no. 11, pp. 237-247, 2011.
[19] Yuanfang Dong, Xiongfei Li, Haiying Zhao, "SBoost: a new imbalanced data learning algorithm", JCIT: Journal of Convergence Information Technology, vol. 7, no. 1, pp. 238- 244, 2012.
[20] Yoav Freund and Robert Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting”, Journal of Computer and System Sciences, vol.55, no.1, pp.119-139, 1997.
[21] Victor R. Basili, Frank E. McGarry, Rose Pajerski, and Marvin V. Zelkowitz, “Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory,” In Proceedings of the 24th International Conference on Software Engineering (ICSE’2002), Orlando, Florida, pp. 69-79, 2002.
[22] Thomas J. McCabe, “A complexity measure”, IEEE Transactions on Software Engineering, vol.2, no.4, pp.308-320, 1976.
[23] Shyam R. Chidamber, and Chris F. Kemerer, “A metrics suite for object-oriented design”, IEEE Transactions on Software Engineering, vol.20, no.6, pp.476-493, 1994.
[24] http://www.cs.waikato.ac.nz/ml/weka/
[25] Mahesh V. Joshi, “On evaluating performance of classifiers for rare classes”, In Proceedings of the 2nd International Conference on Data Mining, Yorktown Heights, NY, USA, pp.641-644, 2002.
[26] Jin Huang and Charles X. Ling, “Using AUC and accuracy in evaluating learning algorithms”, IEEE Transactions on Knowledge and Data Engineering, vol.17, no.3, pp.299-310, 2005.