An Empirical Evaluation of Adaboost Extensions for Cost-Sensitive Classification

Ankit Desai
Charotar Univerisity of Science And Technology
Changa

P. M. Jadav
Dharmsinh Desai University
Nadiad

ABSTRACT

Classification is a data mining technique used to predict group membership for data instances. Cost-sensitive classifier is relatively new field of research in the data mining and machine learning communities. They are basically used for classification tasks under the cost-based model, unlike the error-based model. Error based classifier AdaBoost is a simple algorithm that reweights the training instances to build multiple classifiers in training phase, without considering the cost of misclassification. Out of all generated classifiers in training, in classification, it collects the weighted votes from each and classifies the new sample (example) according to maximum votes collected. Intuitively, combining multiple models shall give more robust predictions than a single model under the situation where misclassification costs are considered. Boosting has been shown to be an effective method of combining multiple models in order to enhance the predictive accuracy of a single model. Thus, it is natural to think that boosting might also reduce the misclassification costs. All the cost-sensitive boosters are studied and five new extensions are proposed and their results are compared in this paper. A few future extensions are notified.

General Terms

AdaBoost, Cost-sensitive classifiers, Data-Mining, Misclassification cost.

Keywords

AdaBoost, Cost-sensitive classifiers, CSExtension1, CSExtension2, CSExtension3, CSExtension4, CSExtension5 Misclassification cost, number of high cost errors.

1. INTRODUCTION

Several classification models have been proposed over the years, e.g. neural networks, statistical models like linear/quadratic discriminates, nearest neighbors, Bayesian methods, Decision trees and Meta learners. Cost-sensitive classifiers are Meta learners to make its base classifier cost-sensitive. Moreover, many classifiers are studied under the error based frame work, which concentrates on improving the accuracy of the classifier. On the other hand, the cost of misclassification is also an important parameter to consider in many applications of classification, such as, credit card fraud detection, medical diagnosis etc. All the error-based classifier methods consider the classification errors as equally likely, which is not the case in all the real-time applications. For example, cost of classifying a credit card transaction as non-fraudulent in case it is actually a fraudulent is quite higher than the cost of classifying a non-fraudulent transaction as a fraudulent. It is recommended that this misclassification costs has to be incorporated into classifier model while dealing with such cost-sensitive applications.

Learning with Cost-Sensitive (CS) algorithms is an essential cluster because it considers cost of misclassification into account apart from basic measures like accuracy and speed. Two ways exists, to incorporate the misclassification cost. First, cost-sensitive classifiers, the algorithm which incorporate misclassification cost into its algorithmic steps. Second, incorporate cost into preprocessing step to make preprocessing stage cost-sensitive e.g. cost-sensitive attribute selection, can be used to incorporate the misclassification cost in pre-processing stage. In case of first type, e.g. AdaBoost and AdaCost, the Meta classifiers are extended to incorporate the cost of misclassification in weight update method. [7]

Many ways have been proposed to incorporate the cost of misclassification. Among all these AdaBoost (Meta learner) and decision tree method has been proven easy to implement and efficient as compared to other methods. Reweighting [4] the training instances by incorporating the cost of misclassification help us to build the models while considering the cost of misclassification.

Intuitively combining multiple models shall give more robust predictions than a single model under the situation where misclassification costs are considered. Boosting has been shown to be an effective method of combining multiple models in order to enhance the predictive accuracy of a single model. Thus, it is natural to think that boosting might also reduce the misclassification costs.

This paragraph will answer why exactly we require cost based models for classification and what is cost matrix with respect to ROC and lift? [17, 18] All error based learners are special case of cost base learners, being cost matrix input [0 1; 0 1]. Lift is used to measure the degree to which the predictions of a classification model over randomly-generated predictions. In classification model, Receiver Operating Characteristic (ROC) is another measure for comparing predicted and original target values. Comparing ROC with lift; lift only applies to binary classification and requires the description of a positive class. Decision making ability of a model can be insight by ROC. How prone is the model to accurately predict the negative or the positive class? Impact of the changes on probability threshold is measured by ROC. The probability threshold is the value (decision point), used by the model for classification. The default probability threshold for binary classification is 0.5. (In multi-class classification, the predicted class is the one predicted with the highest probability.) Discriminating ability of a binary classification model is measured by the area under the ROC curve (AUC) [20]. ROC can be used to find the probability thresholds that yield the highest overall accuracy or the highest per-class accuracy. For example, this is sometimes used to bias decision of prediction over positive or negative class. Probability threshold of a model could easily be changed by a cost matrix. Moreover, Error based models do not answer which model out of many is most advantageous. This can be identified by cost based model (by considering the cost of prediction)[12].

This paper is organized in following sections. i.e. section 2 explains all existing algorithms of the type cost-sensitive boosting. section 3 shows the proposed extensions. Section 4 shows empirical evaluation of all the algorithms. At the end Section 5 concludes the paper.

2. DIFFERENT COST-SENSITIVE BOOSTING ALGORITHMS

K. Ting & Z. Zheng [9] depicted in their research that, Boosting trees has been proven an effective method of reducing the number of high cost errors as well as the total misclassification cost. Moreover, two boosting techniques to incorporate the misclassification cost. Both variant performs significantly better than its predecessor method of boosting i.e. C4.5C. [2] The important thing to note is that both algorithms incorporates the cost of misclassification in post-processing or decision tree induction stage. They have nicely answered the question that why boosting is considered more powerful than other techniques in cost-sensitive classification. Intuitively, combining multiple models shall give more robust predictions than a single model under the situation where misclassification costs are considered. Boosting has been shown to be an effective method of combining multiple models in order to enhance the predictive accuracy of a single model. Thus, it is natural to think that boosting might also reduce the misclassification costs of C4.5c.

Boosting in simple words it is boosting with decision trees but as it utilizes the misclassification cost at minimum expected cost criteria while classification thus boosting becomes cost-sensitive.

Cost-Boosting weights are updated according to misclassification cost associated with each sample. All trees are cost-sensitive. Moreover, weight update rule is as follows:

$$W_{(t+1)}(x) = nw_{(t+1)}(x) / \sum_{y=1}^{N} nw_{(t+1)}(y) N_y$$

$$W_{(t+1)}(x) = \text{Misclassification Cost of i,}$$
$$\text{when sample is assigned as class j}$$
$$\text{else old weight if classified correctly}$$

Weights are normalized to ensure total weight is N.

The only difference is that Cost-Boosting incorporates the misclassification cost during induction of the trees. It also considers the minimum expected cost criteria of boosting while classification of an example.

Discussion: Boosting technique with respect to cost-boosting performs slightly down the line. It is natural because the trees generated in cost-boosting are cost-sensitive (except the first one) so in terms of reduced misclassification cost and reduced number of high cost errors cost-boosting is better. The only disadvantage of the cost-boosting is that in case of change in misclassification cost all the trees have to be regenerated, as it incorporates the cost of misclassification in tree building stage. Further, K. Ting & Z. Zheng [10] added in to their later research viz. UBoost and Cost-Uboost. Moreover, it is important to notice that a slight variation in weight update rule of previous proposal has made UBoost and Cost-UBoost work better in terms of total misclassificaiton cost and number of high cost errors.

Uboost (Boosting with unequal initial weights): this algorithm is similar to the AdaCost algorithm.

w1 (n) = wj = Cj (N / ∑I Ci Ni) is the initial weight update rule.

Misclassification cost incorporated only in first induction of decision tree and at classification stage. Initially unequal weights and minimum expected cost criteria makes this algorithm cost-sensitive.

Cost-UBoost (UBoost with Cost-sensitive adaption): weights are updated according to misclassification cost associated with each sample. All boosted trees are cost-sensitive except the first one. It uses the weight initialization rule of UBoost to start with boosting. A minimum expected cost criterion is used at classification stage. Weights are normalized to ensure total weight is N [19].

Weight update rule:

$$W_{(t+1)}(x) = nw_{(t+1)}(x) / \sum_{y=1}^{N} nw_{(t+1)}(y) N_y$$

$$W_{(t+1)}(x) = \text{Misclassification Cost of i,}$$
$$\text{when sample is assigned as class j}$$
$$\text{else old weight if classified correctly}$$

The only difference is that Cost-UBoosting incorporates the misclassification cost during induction of the trees. It also considers the minimum expected cost criteria of boosting while classification of an example.

Discussion: UBoost technique with respect to Cost-UBoost performs slightly down the line. It is natural because the trees generated in Cost-UBoost are cost-sensitive (except the first one) so in terms of reduced misclassification cost and reduced number of high cost errors Cost-UBoost is better. The only disadvantage of the Cost-UBoost is that in case of change in misclassification cost all the trees have to be regenerated, as it incorporates the cost of misclassification in tree building stage. If we compare the unequal initial weight methods with equal initial weight methods than from the results it can be seen that to start with unequal initial weight is better than its competitor methods.

Wie Fan, Salvatore J. Stolfo, Junxin Zhang and Philip k. chan, in their paper of AdaCost [11], which is based on the intuition that in addition to assigning high initial weights to costly examples like in case of Ting’s Boosting, Cost-Boosting, UBoost and Cost-UBoost, the weight updating rule should also take cost into account and increase the weights of costly misclassification more but decrease the weights of costly correct classification less. The most important thing discussed, explained and proved in this paper is choice of cost adjustment function ß and the hypothesis weight as to reduce this upper bound. They explain the proof the choice for a and ß both. Following comparison shows the difference between original AdaBoost and AdaCost. AdaCost incorporates the cost as a part of algorithmic step.

Table 1. AdaCost modifies the discrete AdaBoost as follows

Descrite AdaBoost	AdaCost
1. Construct – Classifier is constructed using current weights of samples. Confidence level of prediction is saved. 2. Classify $r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)$ $\delta = -1$ if $H_k(x_n) \neq y_n$ $\delta = +1$ otherwise. $\alpha_k = \frac{1}{2}\ln(\frac{1+r_k}{1-r_k})$ 3. Update weight $w_{(k+1)}(n) = w_k(n)\exp(-\delta\mathcal{H}_k(x_n)\alpha_k)$	1. Construct – Classifier is constructed using current weights of samples. Confidence level of prediction is saved. 2. Classify $r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)\beta_\delta$ $\beta_+ = -0.5c_n + 0.5 and \beta_- = 0.5c_n + 0.5$ 3. Update weight $w_{(k+1)}(n) = w_k(n)\exp(-\delta\mathcal{H}_k(x_n)\beta_\delta\alpha_k)$

Table 2. Summarized comparison of cost-sensitive boosting algorithms

Algorithm	Initial Weights	Base Classifier	Which Trees are Cost-sensitive?	Voting Scheme	Weight update equation used?#
Boosting	1/N (Equal)	Decision Tree	No Trees	MECC@	No
Cost-Boost	1/N (Equal)	Decision Tree	All Trees Except the first one	MECC@	Yes
Uboost	(Unequal) *	Decision Tree	No Trees	MECC@	No
Cost-Uboost	(Unequal) *	Decision Tree	All Trees Except the first one	MECC@	Yes

* = Initial Weights: $w_1(n) = w^j = c^j (N / \sum_I C^i N^i)$
# = weight update rules: new weight = misclassification cost * old weight (if incorrectly classified); new weight = old weight (if correctly classified.)
@ = MECC = Minimum Expected Cost Criteria

3. COST SENSITIVE EXTENSIONS (CSEXTIONSIONS)

3.1 CSExtension1

AdaBoost is addressed in this algorithm. these evaluate cost during training process. The simplest is the only variant that does not use confidence rated predications in the weight update rule as in step iii as indicated in following algorithm. This variant can be obtained as follow.

Algorithm:

Given a training set T containing N examples (x_n,y_n) where x_n is vector of attribute values and y_n Y is the class label, $w_k(n)$ denotes the weight of the nth example at the kth trial.

Initialization: $w_1(n) = 1$.

In each trial k = 1,…..,K the following three steps are carried out.

i. A model H_k is constructed using the training set under the weight distribution w_k. let $H_k(x)$ denotes the predicted class, and $H_k(x) \in [0, 1]$ denote the confidence level of the prediction.

Classify T using H_k and compute,

$$r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)$$

Where,

$\delta = -1$ if $H_k(x_n) \neq y_n$ and $\delta = +1$ otherwise.

Then compute,

$$\alpha_k = \frac{1}{2}\ln(\frac{1+r_k}{1-r_k})$$

ii. The weight vector $w_{(k+1)}$ for next trial is updated as follows:

$$w_{(k+1)}(n) = C_\delta w_k(n)$$

Where, C_δ = cost of classification.

After K models are included, they are ready for prediction. For each classification, the final prediction is combined prediction from the K models using the maximum vote criterion, computed as follow.

$$H^*(x) = \arg\max_{y \in Y} \sum_{k:H_k(x)=y} \alpha_k \mathcal{H}_k(x)$$

3.2 CSExtension2

AdaBoost is addressed in this algorithm. these evaluate cost during training process. Another variant is proposed to study the effects of including cost and another parameter α in weight update rule. This variant does not use α in weight update- rule. This variant can be obtained as follow.

Given a training set T containing N examples (x_n,y_n) where x_n is vector of attribute values and $y_n \in Y$ is the class label, $w_k(n)$ denotes the weight of the nth example at the kth trial.
Initialization: $w_1(n) = 1$.

In each trial k = 1,…..,K the following three steps are carried out.

i. A model H_k is constructed using the training set under the weight distribution w_k. let $H_k(x)$ denotes the predicted class, and $\mathcal{H}_k(x) \in [0, 1]$ denote the confidence level of the prediction.
ii. Classify T using H_k and compute,

$$r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)$$

Where,

$\delta = -1$ if $H_k(x_n) \neq y_n$ and $\delta = +1$ otherwise.

Then compute,

$$\alpha_k = \frac{1}{2}\ln(\frac{1+r_k}{1-r_k})$$

iii. The weight vector $w_{(k+1)}$ for next trial is updated as follows:

$$w_{(k+1)}(n) = C_\delta w_k(n)\exp(-\delta\mathcal{H}_k(x_n))$$

Where, C_δ = cost of classification.

After K models are included, they are ready for prediction. For each classification, the final prediction is combined prediction from the K models using the maximum vote criterion, computed as follow.

$$H^*(x) = \arg\max_{y\in Y} \sum_{k:H_k(x)=y} \alpha_k \mathcal{H}_k(x)$$

3.3 CSExtension3

AdaBoost is addressed in this algorithm. these evaluate cost during training process. Another variant is proposed to study the effects of including cost and another parameter α in weight update rule. This variant uses α in weight update- rule. This variant can be obtained as follow.

Given a training set T containing N examples (x_n,y_n) where x_n is vector of attribute values and $y_n \in Y$ is the class label, $w_k(n)$ denotes the weight of the nth example at the kth trial.
Initialization: $w_1(n) = 1$.
In each trial k = 1,…..,K the following three steps are carried out.

i. A model H_k is constructed using the training set under the weight distribution w_k. let $H_k(x)$ denotes the predicted class, and $\mathcal{H}_k(x) \in [0, 1]$ denote the confidence level of the prediction.
ii. Classify T using H_k and compute,

$$r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)$$

$$\alpha_k = \frac{1}{2}\ln(\frac{1+r_k}{1-r_k})$$

Where,

$\delta = -1$ if $H_k(x_n) \neq y_n$ and $\delta = +1$ otherwise.

Then compute,

$$\alpha_k = \frac{1}{2}\ln(\frac{1+r_k}{1-r_k})$$

iii. The weight vector $w_{(k+1)}$ for next trial is updated as follows:

$$w_{(k+1)}(n) = C_\delta w_k(n)\exp(-\delta\mathcal{H}_k(x_n)\alpha_k)$$

Where, C_δ = cost of classification.

After K models are included, they are ready for prediction. For each classification, the final prediction is combined prediction from the K models using the maximum vote criterion, computed as follow.

$$H^*(x) = \arg\max_{y\in Y} \sum_{k:H_k(x)=y} \alpha_k \mathcal{H}_k(x)$$

Further, we require that β_+ is non-increasing with respect to c_n. This means that the reward for correct classification is low when the cost is high, and vice versa. This seems to be counter-intuitive, and it could be the source of AdaCost's poor performance. Based on this fact we proposed another two proposals namely CSExtension4 and CSExtension5. They are elaborated as follow.

We alter the form of β_δ and assume β_+ is non-decreasing as is β_-, and they are both equal to the cost of misclassification, that is, $\beta_+ = \beta_- = c_n$ and denote the resultant algorithm as CSExtension4. We, also employ a second modification, called CSExtension5, which is identical to CSExtension4 except that β_δ is excluded in r_k equation. This is to investigate whether cost needs to be taken into consideration in step (ii) of the boosting procedure.

3.4 CSExtension4

Given a training set T containing N examples (x_n,y_n) where x_n is vector of attribute values and $y_n \in Y$ is the class label, $w_k(n)$ denotes the weight of the nth example at the kth trial. In each trial k = 1,…..,K the following three steps are carried out.

i. A model H_k is constructed using the training set under the weight distribution w_k. let $H_k(x)$ denotes the predicted class, and $\mathcal{H}_k(x) \in [0, 1]$ denote the confidence level of the prediction.
ii. Classify using H_k and compute,

$$r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)\beta_\delta$$

$$\alpha_k = \frac{1}{2}\ln(\frac{1+r_k}{1-r_k})$$

Where, $\beta_+ = \beta_- = c_n$

iii. The weight vector $w_{(k+1)}$ for next trial is updated as follows:

$$w_{(k+1)}(n) = w_k(n)\exp(-\delta\mathcal{H}_k(x_n)\beta_\delta\alpha_k)$$

After K models are included, they are ready for prediction. For each classification, the final prediction is combined prediction from the K models using the maximum vote criterion, computed as follow.

$$H^*(x) = \arg\max_{y\in Y} \sum_{k:H_k(x)=y} \alpha_k \mathcal{H}_k(x)$$

3.5 CSExtension5

Given a training set T containing N examples (x_n,y_n) where x_n is vector of attribute values and $y_n \in Y$ is the class label, $w_k(n)$ denotes the weight of the nth example at the kth trial.
In each trial k = 1,…..,K the following three steps are carried out.

i. A model H_k is constructed using the training set under the weight distribution w_k. let $H_k(x)$ denotes the predicted class, and $\mathcal{H}_k(x) \in [0, 1]$ denote the confidence level of the prediction.
ii. Classify using H_k and compute,

$$r_k = \frac{1}{N}\sum_n \delta w_k(n)\mathcal{H}_k(x_n)$$

$$\alpha_k = \frac{1}{2}\ln\left(\frac{1+r_k}{1-r_k}\right)$$

Where, $\beta_+ = -0.5c_n + 0.5 and \beta_- = 0.5c_n + 0.5$

iii. The weight vector $w_{(k+1)}$ for next trial is updated as follows:

$$w_{(k+1)}(n) = w_k(n)\exp(-\delta\mathcal{H}_k(x_n)\beta_\delta\alpha_k)$$

After K models are included, they are ready for prediction. For each classification, the final prediction is combined prediction from the K models using the maximum vote criterion, computed as follow.

$$H^*(x) = \arg\max_{y\in Y} \sum_{k:H_k(x)=y} \alpha_k \mathcal{H}_k(x)$$

Note: All the algorithms proposed here can use the minimum expected cost criterion by implementation of the following equation.

$$H^*(x) = \arg\max_{j} \sum \sum_{k:H_k(x)=y} \alpha_k \mathcal{H}_k(x)\cos t(i,j)$$

4. EMPIRICAL EVALUATION

In this section, we empirically evaluate the performances of the cost-sensitive boosting procedures (Firstly, AdaBoost, Boost, Cost-Boost, UBoost and Cost-UBoost with AdaCost. Secondly, AdaBoost, CSExtension1, CS Extension2, CSExtension3, CSExtension4 and CSExtension5 with AdaCost). We have divided experiments in two groups because there are in total eleven algorithms so to reduce the complexity perform algorithms in two groups. Where the first group is having the algorithms which use the boosting concept and voting criteria to and differs as per table 2. Whereas, the second group is having the algorithms which modifies the weight update equation. AdaCost and AdaBoost are in both the groups to provide mapping and comparisons.

Fourteen two-class natural data sets from the UCI machine learning repository are used in the experiments [13-14]. They are breast cancer (Wisconsin), liver disorder, credit screening, echocardiogram, solar flare, heart disease (Cleveland), hepatitis, horse colic, house-voting 84, hypothyroid, king-rook versus king-pawn, pima Indian diabetes, sonar and tic-tac-toe data sets. Only two-class data sets are used because AdaBoost, in its basic form, is designed for two-class problems only.

For each of the data sets, we report the sum of six averages, where each average is the result of a run of two 10-fold cross-validations using a fixed cost factor. The six cost matrices used are [0 1; 2 0], [0 2; 1 0], [0 1; 5 0], [0 5; 1 0], [0 10; 1 0] and [0 1; 10 0]. Thus, in each set of experiments (having six cost factors and fourteen data sets), there are altogether 84 runs.

The key measure to evaluate the performance of the algorithms for cost-sensitive classification is the total cost of misclassifications made by a classifier on a test set (i.e., $\sum_m$ = *cost*(*actual*(*m*), *predicted*(*m*))). In addition, we also use a second measure: the *number of high cost errors*. [5-6], [15-16] It is the number of misclassifications associated with costs higher than unity made by a classifier on a test set. The second measure is used primarily to explain the behavior of the algorithms.[1]

To conduct the experiments described in the experiment section, J48 underlies the model H_k in the boosting procedure (J48 is used as a base classifier). Only the default settings of J48 are used. In our implementation all variants incorporate the minimum expected cost criterion. Moreover, a weight initialization process is used defined in following equation. where, C(i) be the cost of misclassifying a class i instance; the weight of a class i instance can be computed as, such that the sum of all instance weights is $\sum_i (i) N_i = N$.

$$w_{(k)}(i) = C(i)\frac{N}{\sum_j C(j)N_j}$$

– **Performance of Boost, Cost-Boost, UBoost, Cost-UBoost and AdaCost**

1. Experiment for calculation of total misclassification costs

Graph 3. shows the results of total cost of misclassification for all selected datasets. From graph 3 and graph 1 it is clear that AdaCost outperforms all competitors in terms of total misclassification cost.

– **Observation:** By performing the above experiment we were intend to find out that which existing cost-sensitive booster performs better in terms of total misclassification cost.

2. Experiment for calculation of number of high cost errors

Graph 4 shows the results of total number of high cost errors for all selected datasets. From graph 4 and graph 2 it is clear that AdaCost outperforms all competitors in terms of number of high cost errors.

– **Observation:** By performing the above experiment we were intend to find out that which existing cost-sensitive booster performs better in terms of number of high cost errors. From the above experiments we can conclude that AdaCost wins in terms of both number of high cost errors as well as total cost of misclassification.

– **AdaBoost, CSExtension1, CSExtension2 and CSExtension3, CSExtension4 and CSExtension5 with AdaCost**

1. Experiment for calculation of total misclassification costs

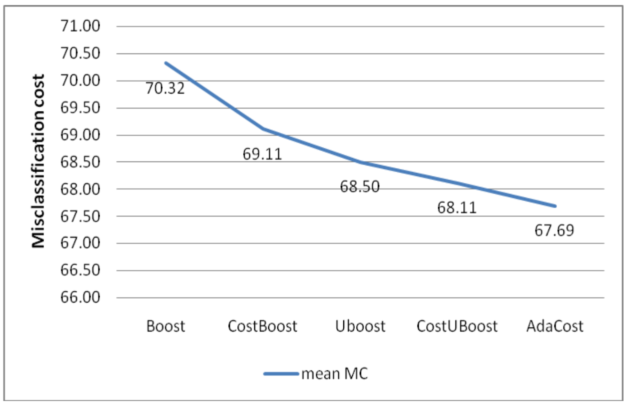
Graph 7. shows the results of total cost of misclassification for all selected datasets. From graph 7 and graph 5 it is clear that CSExtension3 outperforms all competitors in terms of total misclassification cost.

– **Observation:** By performing the above experiment we were intend to find out that which existing cost-sensitive booster performs better in terms of total misclassification cost.

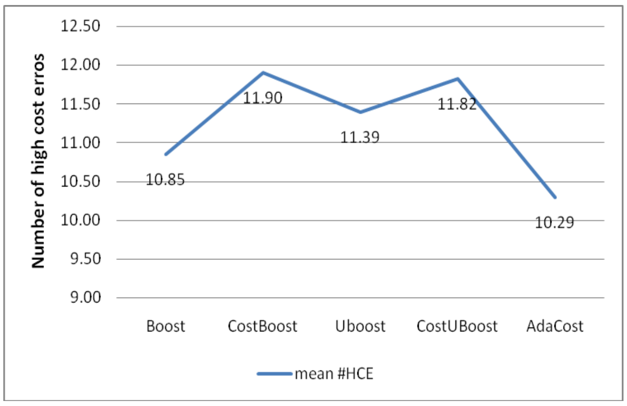
2. Experiment for calculation of number of high cost errors

Graph 8 shows the results of total number of high cost errors for all selected datasets. From graph 8 and graph 6 it is clear that CSExtension2 outperforms all competitors in terms of number of high cost errors.

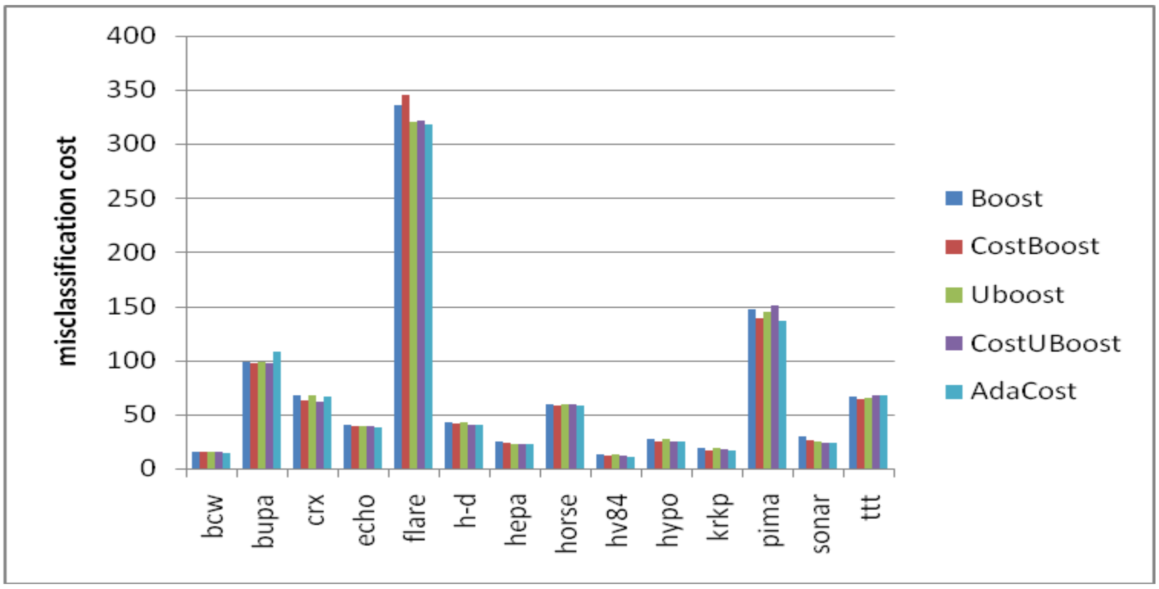
– **Observation:** By performing the above experiment we were intend to find out that which existing cost-sensitive booster performs better in terms of number of high cost errors. From the above experiments we can conclude that CSExtension3 wins in terms of total misclassification cost and CSExtension2 in terms of number of high cost errors.



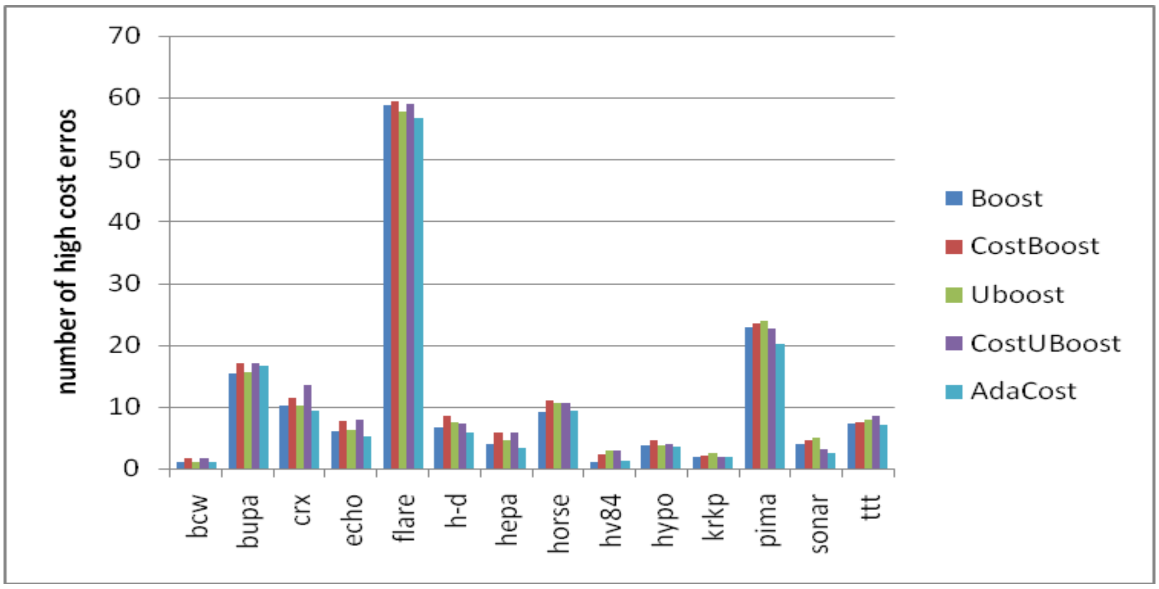
Graph 1. Average misclassification cost of Boost, CostBoost, Uboost, CostUBoost and AdaCost



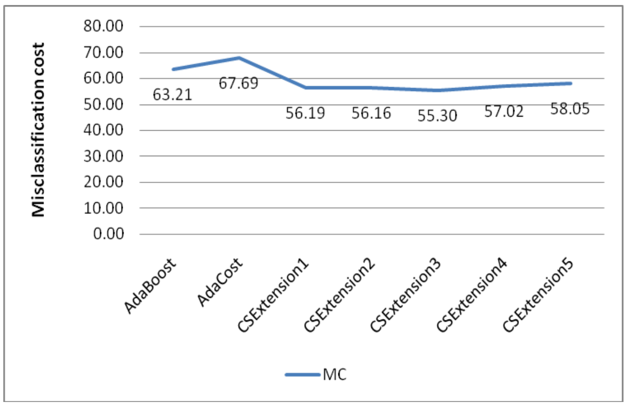
Graph 2. Average number of high cost errors cost of Boost, CostBoost, Uboost, CostUBoost and AdaCost



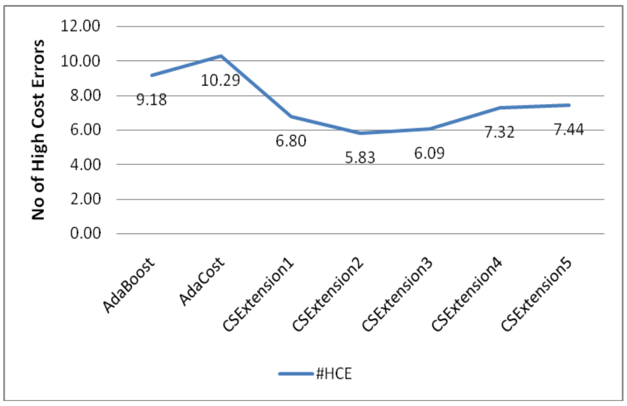
Graph 3.Total misclassification cost of Boost, Cost-Boost, UBoost, Cost-UBoost and AdaCost



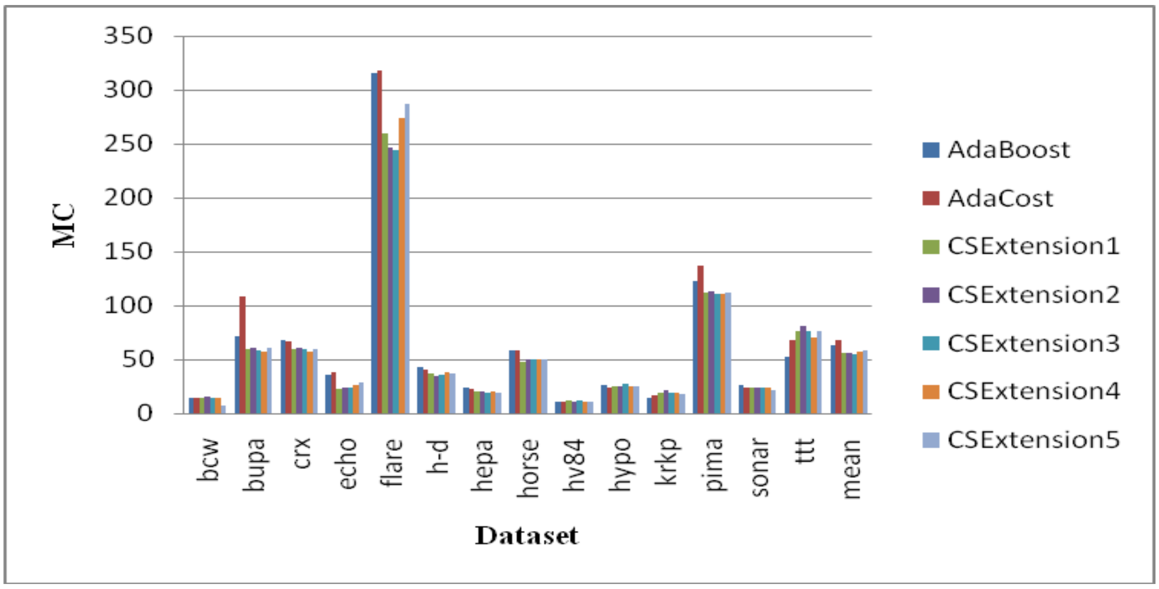
Graph 4. Number of high cost errors of Boost, Cost-Boost, UBoost, Cost-UBoost and AdaCost



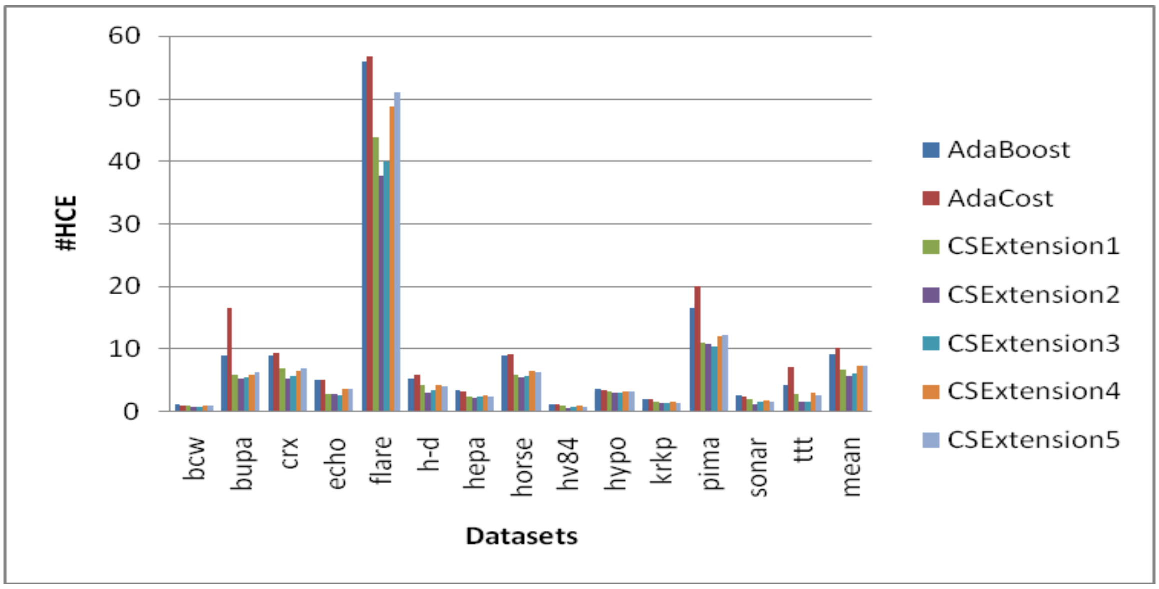
Graph 5. Average misclassification cost of AdaBoost, AdaCost, CSE1, CSE2, CSE3, CSE4 and CSE5



Graph 6. Average number of high cost errors of AdaBoost, AdaCost, CSE1, CSE2, CSE3, CSE4 and CSE5



Graph 7.Total misclassification cost of AdaBoost, AdaCost, CSExtension1, CSExtension2, CSExtension3, CSExtension4 and CSExtension5



Graph 8. Number of high cost errors of AdaBoost AdaBoost, AdaCost, CSExtension1, CSExtension2, CSExtension3, CSExtension4 and CSExtension5

5. CONCLUSION

This paper provides detail discussion on the algorithms of cost-sensitive boosting. At the best of our knowledge we believe, all the cost-sensitive boosters are studied.

Moreover, we believe that the comparative analysis, summarized analysis depicted here is useful for many researchers in their work for cost-sensitive boosting.

Emparical evaluation justified that as per Wie Fan, Salvatore J. Stolfo, Junxin Zhang and Philip k. chan, have listed in their paper of AdaCost, that AdaCost is the best algorithm available for the type of data that we have selected to reduce number of high cost errors and total cost of misclassification. The reason for its better performance is that it incorporates the misclassification cost as a part of its algorithmic step as well as the selection of the parameter β in weight update rule plays a vital role in reduction of the cost and number of high cost errors.

Issues listed below in existing system leaves the wide scope of improvments possible in the existing algorithms.

– Moreover, from [3] we can get an idea of how exactly the confidence rated predictions help us increase the classification accuracy.

– There is no comparative study which shows the effects of minimum expected cost criterion versus maximum vote criterion.

– In all the Meta classifiers analyzed above decision tree is chosen as a base learner. One can further show the comparative study of choosing Ibk, neural networks, naïve base classifier or other learners as base learners.

6. REFERENCES

[1] Tao wang, Zhenxing Qin, Zhi Jin and Shichao Zhang , "*Handling overfitting in test cost-sensitive decision tree learning by feature selection, smoothing and pruning*", The journal of systems and software, 2010.

[2] Susan Lomax and Sunil Vadera, "*An empirical comparison of cost-sensitive decision tree induction algorithms*", july 2011.

[3] Schapire and Singer, "*Improved boosting algorithms using confidence-rated predictions*". Machine learning, 1999.

[4] Bianca Zadrozny, John Langford, Naoki Abe, "*Cost-Sensitive Learning by Cost-Proportionate Example Weighting*", Proceedings of the Third IEEE International Conference on Data Mining (ICDM’03).

[5] Geoffrey I. Webb, "*Cost-Sensitive Specialization*", Proceedings of the 1996 Pacific Rim International Conference on Artificial Intelligence, Cairns, Springer-Verlag, pp. 23-34.

[6] Alan T. Remaley, Maureen L. Sampson, James M, Deleo, Nancy A. Remaley, Beriuse D. Farsi and Mark H. Zweig, "*Prevalence-Value-Accuracy Plot: A new method for comparing diagnostic tests based on misclassification costs*", 1999.

[7] P. Domingos. "*Metacost: A general method for making classifiers cost-sensitive*", In KDD, pages 155–164, 1999.

[8] Artur Ferreira, "*Survey on boosting algorithms for supervised and semi-supervised learn*ing", oct. 2007.

[9] Kai ming Ting and Zijian Zheng, "*Boosting Cost-sensitive trees*", Tenth International Conference on Discovery Science, LNAI-1038 (pp.134-145) Japan: Springer- 2007.

[10] Kai Ming Ting and Zijian Zheng, "*Boosting Trees for cost-sensitive classificaiton*" , Eighth International Conference on DS, Singapore 2005.

[11] Wie Fan, Salvatore J. Stolfo, Junxin Zhang and Philip k. chan "*AdaCost: Misclassification Cost-sensitive Boosting*", 27th International Conference on Machine Learning, july 2010.

[12] Web link: Oracle® Data Mining Concepts 11g Release 1, http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/classify.html

[13] Data-set downloaded from: http://tunedit.org/repo/UCI/credit-a.arff

[14] UCI machine learning repository for dataset: http://archive.ics.uci.edu/ml/datasets.html

[15] Wiki pedia page for Prevalence: http://en.wikipedia.org/wiki/Prevalence, updated on 5 July 2011.

[16] Wiki pedia page for sensitivity and specificity: http://en.wikipedia.org/wiki/Sensitivity_and_specificity, updated on 19 July 2011.

[17] Tutorial on DB2 business intelligence, article on cost matrix, http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.im.model.doc/c_cost_matrix.html

[18] Tutorial on Oracle® Data Mining Concepts 11g Release 1, http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/classify.htm

[19] OAIDTB: Boosting extensions for WEKA, web-link: http://pisuerga.inf.ubu.es/lsi/Software/oaidtb/

[20] Books: Data mining - Concept and Techniques by Han & Kamber. Data mining: concepts and techniques. The Morgan Kaufmann series in data management systems, ISBN- 1558609016, 9781558609013, publisher morgan, 2006.