Building Cost-Sensitive ELEM2

Ossama Abdel-Hamid and Zahidur Rahman

Department of Computer Science and Engineering,
York University
Toronto, ON M3J1P3, Canada
e-mail:{ossama,zahidur}@cse.yorku.ca
http://www.cse.yorku.ca/{~ossama,~zahidur}

Abstract. Most of classification learning methods aim at the reduction of the number of errors. However, in many real-life applications it is misclassification cost, which should be minimized. In this paper we are integrating cost-sensitivity to ELEM2. Our approach is to change the existing system ELEM2 to cost sensitive decision rue learning. First, we applied Instance weighting to ELEM2. We changed the whole structure of it to fit with the Instance weighting method. Then we proposed a new method in its post pruning step, without changing the remaining components of ELEM2 i.e., the classification gain and significant value are not changed. Finally, the performance of our method is compared to that of ELEM2 without any cost methods and ELEM2 with instance weighting method. The results show, that our new method is able to reduce the misclassification cost much better then with instance weighting.

1. Introduction

Classification techniques are aimed at building models from predefined instance classes based on a training set of data instances with known class labels. This model can be used to predict classes of future unclassified data instances. There exist many effective methods for building classifiers [3], but in most cases the goal is to minimize the number of misclassification errors. However, in many real life applications the assumption that all errors are equally important is invalid. For example, in medical domain misclassifying an ill patient as a healthy one is usually much more harmful than treating a healthy patient as an ill one and sending him for additional examinations. In database marketing the cost of mailing to a non-respondent is very small, but the cost of not mailing to someone who would respond is the entire profit lost [4].

While most basic classification methods have not been designed to solve cost-sensitive problems, two major group of works have been proposed for converting these methods in to cost sensitive classifier learners, *internal* and

external methods. *External* methods are general methods that can be used to convert any error-based classifiers into cost sensitive ones. These methods treat the classifier algorithms as black-boxes without need to have any knowledge of the structure of them. So, the internal structure of the classifiers is untouched, wrapping a external method around them. Most famous proposed external methods are *MetaCost*[4] and *stratification*[5.6,7]. On the other hand, *Internal* methods are most commonly used approach, to apply them to any error-based classifiers, internal structure of classifiers should be known and need to change the whole structure of the classifiers to fit the used *internal method*. Two major *internal* approaches, *instance weighting* [4,8,9,10] and *post hoc threshold adjusting* [11,12], have been proposed for converting classifiers into cost-sensitive ones.

Although the external methods work fine in some fields, the best approach is to modify the internal structure of any classifier to be cost-sensitive. Zhao [13] has studied the comparison between two internal methods, *instance weighting* and *threshold adjusting* and concluded that for continuous probability estimates, such as Naive Bayes, logistic regression, and backpropagation neural network *post hoc threshold adjusting* works fine, but for the symbolic classification methods, such as decision trees, decision rule, and decision table learners, *instance weighting* are very suitable for them.

In our work, we modified the existing system ELEM2 [1], a decision rule learning, according to the instance weighting and also produced our own approach to build cost-sensitive ELEM2. We then compared the result of both and conclude with an important result.

In the reminder of this paper, we first introduce some relative works done in the same field. Next we discuss the instance weighting as one of the internal methods and how to modify the ELEM2 to fit the instant weighting method. Then we produced our own solution to build Cost-sensitive ELEM2 during post-pruning process. Finally, we will end up with experiment results, outcome of these results and future works that should be done in this field.

2. Related work

The process of inductive learning may involve different costs [14] e.g., costs of tests (features), costs of cases, costs of errors. In the literature the latter kind of costs is the most commonly discussed one.

Several attempts to incorporate misclassification costs into decision tree or decision rule learning were made so far. The first approach was introduced by

Breiman et al. [15] in CART decision tree learning system. Their method consists in modification of the class prior probabilities used in the splitting criterion. The cost-based measure is also used for tree pruning.

In a simpler approach (e.g., [16], [17]) error costs are taken into consideration during the pruning phase, but not during the induction phase. In such case the pruning procedure has a limited capability to change the structure of the classifier obtained by the error-based learning. Consequently, ignoring the misclassification cost at the first phase is the main drawback of this approach.

Pazzani et al. [18] introduced three cost-sensitive algorithms for decision list induction. Their method was applied to a real telephone network troubleshooting problem.

Ting [19] proposed a modified version of C4.5 using instance-weighting for induction of cost-sensitive decision trees. This approach requires the conversion of the cost matrix into the cost vector, which may result in poor performance in multi-class problems.

In [4] Domingos presented a method for making an arbitrary classifier cost-sensitive by wrapping a cost-minimizing procedure around it. However his approach may be computationally inefficient because it requires many runs of the basic learning algorithm.

3. Instance Weighting

The main idea behind instance weighting is giving instances that belongs to classes which has high rate misclassification cost over those belongs to classes with low rate misclassification cost. To do so instance weighting modifies the weight of an instance proportional to the cost of misclassifying the class to which the instance belongs, leaving the sum of all training instance weights still equal to the total number of training instances *N*. The last condition is important because there is no reason to alter the size of the training set, which is equivalent to the sum of all training instance weights, while the individual instance weights are adjusted to reflect the relative importance of instances for making future prediction with respect to cost-sensitive classification.

Let *C(j)* be the cost of misclassifying a class *j* instance, then the weight of a class *j* instance can be computed as:

$$w(j) = C(j)\frac{N}{\sum_i C(i)N_i} \qquad (1)$$

Such that the sum of all instance weights is

$$\sum_j w(j)N_j = N .$$

For $C(j) \geq 1$, $w(j)$ has the smallest value, $0 < \frac{N}{\sum_i C(i)N_i}$, when $C(j)=1$; and the largest value, $w(j) = C(j)\frac{N}{\sum_i C(i)N_i} > 1$, when $C(j)=max_i C(i)$.

We modified ELEM2 to create ELEM2IW. First, we needed to initialize the training instance weights to $w(j)$. Then we modified the structure of ELEM2, so in every equation the probability measure is no longer the count of instances, instead it is the sum of their weights. We did this modification in significance value function *SIG*, which measures the degree of relevance of an attribute-value pair, Classification gain *CG*, which measures how much is gained by classifying a new example into a class based on the information about the probabilities of a set of attribute-value pairs and the class, and rule quality measure $Q(r)$, which is used as a criterion for post-pruning and in the classification part of ELEM2.

This modification effectively converts the standard ELEM2 rule induction procedure that seeks to minimize the number of errors, regardless of cost, to a procedure that seeks to minimize the number of errors with high cost. Note that minimizing the later does not guarantee that the total misclassification cost is minimized. This is because the number of low cost errors is usually increased as a result. The next topic will describe our proposed approach for minimizing the overall cost.

In a classification task of K classes, the misclassification costs can be specified in a cost matrix of size $K \times K$. The row of the matrix indicates the predicted class, and the column indicates the actual class. The off-diagonal entries contain the costs of misclassifications; and on the diagonal lie the costs for correct classifications which are zero in this case.

Let $cost(i,j)$ be the cost of misclassifying an instance belonging to class j as belonging to class i. In all cases, $cost(i,j) = 0$, for $i = j$. A cost matrix must be converted to a cost vector $C(j)$ in order to use Equation (1) for instance-weighting. In our project, we employ the form of conversion suggested by Breiman *et al.* (1984):

$$C(j) = \sum_{i}^{I} \cos t(i, j) \qquad (2)$$

4. Our Approach

The rule induction in ELEM2 depends on the significance measure. The value of the significance measure depends on how much a certain attribute value pair succeeds to increase the probability of the target class given this attribute value pair. This property should be kept. So, the rule generation step is kept exactly the same. Instead, the pruning step is modified. The target of the modification of pruning step is to generate more general rules for the more important classes which have higher misclassification costs by doing more pruning. While the inverse is done with the less important classes. As a result, the more important class rules have more probability to match with new instances.

The rule quality measure for rule *r* is defined as:

$$Q(r) = \log \frac{P(r \mid c)(1 - P(r \mid \neg c)}{P(r \mid \neg c)(1 - P(r \mid c))} \qquad (3)$$

$$= \log \frac{m(N - n - M + m)}{(n - m)(M - m)} \qquad (4)$$

Where,

m: is the number of positive examples covered by rule *r*
n: is the number of examples covered by rule *r*,
M: is the total number of positive examples, and
N: is the total number of examples in the dataset.

It can be seen that as more attribute value pairs usually *m* gets larger and in the same time *n* gets larger too. So, for a certain attribute value pair, the value of Q(*r*) increases if the increase in m is high compared to the increase in *n*. So if we made m to increase, the value of Q(*r*) increases too allowing more pruning. So, we define the modified value of *m* as:

$$m' = n \frac{w(c)m}{w(c)m + n - m} \qquad (5)$$

Where $w(c)$ is the weight of class c. The effect of this modification is to increase m in proportion to the class weight while keeping it less than or equal to n. Then the quality measure of rule r is defined as:

$$Q(r) = \log \frac{m'(N - n - M + m')}{(n - m')(M - m')} \qquad (6)$$

It should be noted that we should add the condition $M \le m' \le M + n - N$ to make sure that the ratio doesn't go to a negative value.

This modification increases m proportionally to the class weight, which allows more pruning for the higher misclassification cost classes. The same modified value of Q(r) should be used also to rank the rules in order to make the more important class rules have better quality.

We have changed ELEM2 according to the previous explanations and named it ELEM2CS.

5. Experimental results

Four measures are used to evaluate the performance of the ELEM2IW and ELEM2CS. They are total misclassification costs, average misclassification costs, Average accuracy and average number of rules. The first is the most important measure in any cost-sensitive classifications. A good cost-sensitive classifier will have total misclassification costs as low as possible. Everything being equal, a rule induction algorithm is better than the other if it induces smaller average number of rules.

We conduct experiments using ten datasets obtained from the UCI repository of machine learning databases. The datasets are selected to cover a wide variety of different domains with respect to dataset size, the number of classes, the number of attributes, and types of attributes. They consist of five two-class datasets and five multi-class datasets.

Ten 10-fold cross-validations with post-pruning are carried out in each dataset; random cost assignments with the unity condition are used in all datasets, where in each non-diagonal entry in the cost matrix an integer randomly generated between 1 to 10 is assigned. Three experiments were carried out for each dataset with fixed generated cost vector, first with ELEM2, second with ELEM2IW, and the third with ELEM2CS. The collected data is shown in table (1).

Table1. ELEM2, ELEM2IW and ELEM2CS in terms of cost, average cost, average accuracy and average no of rules.

Dataset	Cost Vector	Cost			Average Cost		
		Elem2	Elem2IW	Elem2CS	Elem2	Elem2IW	Elem2CS
a	[3 2]	2550	2271	1588	0.17	0.15	0.11
abalone	[1 5 1]	5857	5485	2538	1.4	1.31	0.61
bcdata	[3 5]	111	103	92	0.16	0.15	0.13
ecoli	[9 7 1 2 6 1 1 8]	207	222	199	0.61	0.66	0.59
example2	[5 1]	0	2	3	0	0.01	0.01
iris	[1 1.5 10]	34.5	24.5	14.5	0.23	0.16	0.097
monks-1	[7 2]	13	18	13	0.1	0.14	0.1
monks-2	[5 1]	161	247	178	0.95	1.46	1.05
optdigits	[10 9 8 7 6 5 4 3 2 1]	1370	1366	1508	0.36	0.36	0.39
wine	[1 1.5 5]	32	35	32	0.18	0.2	0.18
Dataset	Cost Vector	Accuracy			Average No of rules		
		Elem2	Elem2IW	Elem2CS	Elem2	Elem2IW	Elem2CS
a	[3 2]	93.82	94.44	95.92	241.8	226.4	244
abalone	[1 5 1]	55.64	55.16	47.67	297.7	344.7	493.9
bcdata	[3 5]	96.04	96.34	96.19	16.2	16.4	12.4
ecoli	[9 7 1 2 6 1 1 8]	83.96	77.42	81.59	29	24.1	27.6
example2	[5 1]	100	100	100	6.2	6.3	6.2
iris	[1 1.5 10]	96	96.66	97.33	7.3	4.5	8.2
monks-1	[7 2]	96.8	96.8	96.8	9.4	9.1	9.3
monks-2	[5 1]	66.25	58	63.24	42	37	41.5
optdigits	[10 9 8 7 6 5 4 3 2 1]	93.43	93.57	92.13	84.6	84.3	83.8
wine	[1 1.5 5]	94.24	93.72	94.24	7.2	7.2	7.2

From table 1, it can be seen that the proposed modification works much better than instance weighting. The cost always decreases or at least stays the same with the exception of the monks-2 and optsdigits datasets where the cost increased a little. Instance weighting sometimes achieves a lower misclassification costs, but always the gain is little compared to our proposed Elem2CS and surprisingly it sometimes gets much worse results as with monk2-2 dataset.

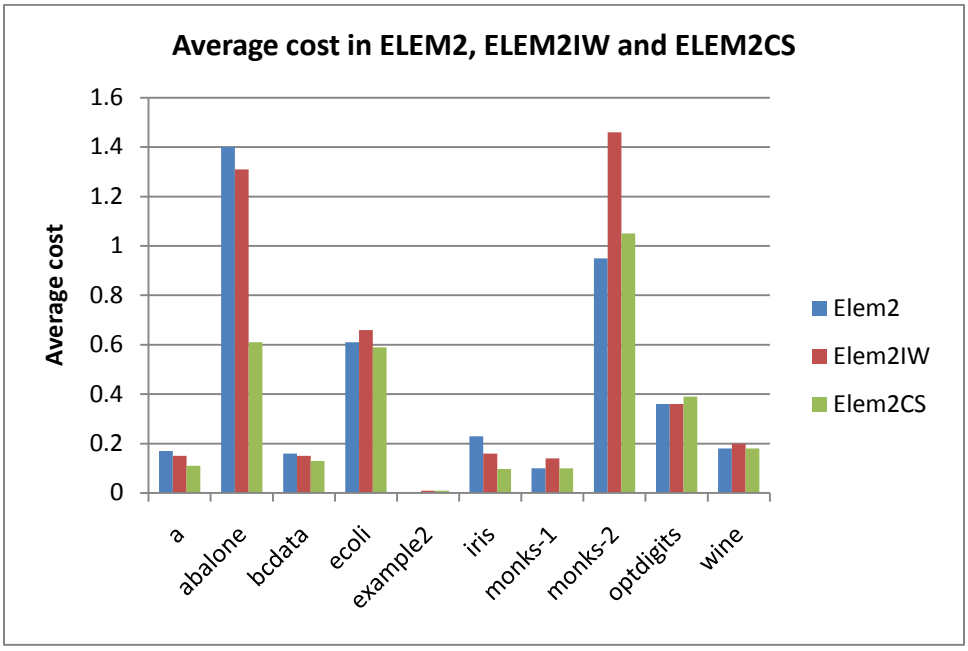

Figure 1

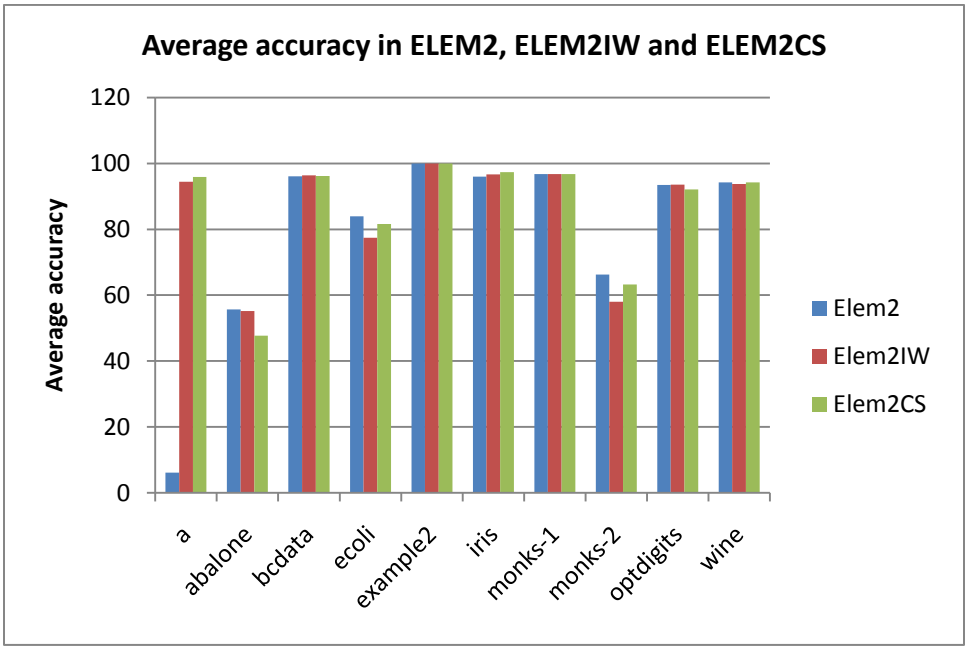

Figure 2

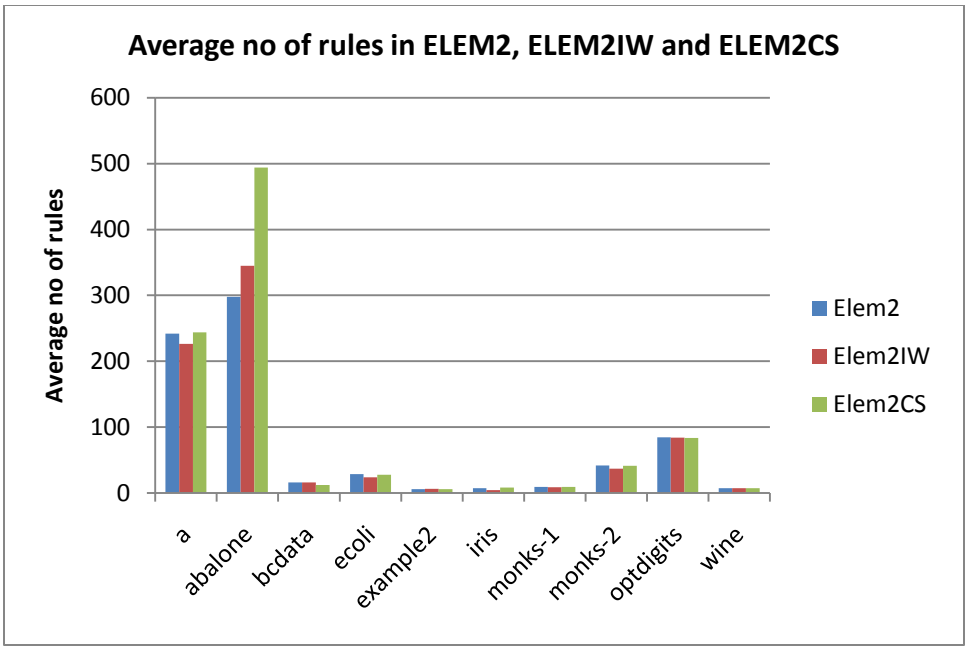

Figure 3

6. Conclusions

In this report, a modified version of ELEM2 is presented that handles the case of cost sensitive classification. Experimental results showed that the proposed modification achieved much better results in respect to minimizing the misclassification cost.

The idea of making more important class rules more general proved to work to favor the classification of the more important classes. And it proved to better and more robust than instance weighting.

Although, more research is needed in order to analyze the performance of the proposed modification to get more insight about when the algorithm performs well and why it fails to decrease the cost with some cases. Also, a comparison between the performance of the proposed algorithm and that of the general (external) cost sensitive algorithms such as meta-cost is needed.

References

1. An, A. Learning Classification Rules from Data, International Journal of Computers and Mathematics with Applications, Vol.45, No.4-5, pp.737-748, 2003.

2. Kwedlo,W.,Kretowski, M.: An Evolutionary Algorithm for Cost-Sensitive Decision Rule Learning, ECML 2001, LNAI 2167, pp. 288-299,2001.

3. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood Ltd. (1994) .

4. Domingos, P.: MetaCost: A general method for making classifiers cost-sensitive. In Proc. of Int. Conf. on Knowledge Discovery and Data Mining, KDD'99. ACM Press (1999) 155-164.

5. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

6. P. Chan and S. Stolfo. Toward scalable learning with non-uniform class and cost distributions. Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining, pp. 164-168, New York, NY, 1998.

7. F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. Proc. 15th Intl. Conf. on Machine Learning, pp. 445-453, Madison, WI, 1998.

8. Margineantu D (2002) Class probability estimation and cost-sensitive classification decisions. In: Proceedings of the 13th European conference on machine learning (ECML), Helsinki, Finland, pp 270–281

9. Ting KM (2002) An instance-weighting method to induce cost-sensitive trees. IEEE Trans Knowl Data Eng 14(3): 659–665

10. Zadrozny B, Langford J, Abe N (2003) Cost-sensitive learning by cost-proportionate example weighting. In: Proceedings of the 3rd IEEE international conference on data mining (ICDM), Melbourne, Florida, pp 435–442

11. Afifi AA, Clark V (1996) Computer-aided multivariate analysis, 3rd edn. Chapman & Hall, London.

12. Sinha AP, May JH (2005) Evaluating and tuning predictive data mining models using receiver operating characteristic curves. J Manage Inf Syst 21(3): 249–280

13.Zhao, Huimin: Instance weighting versus threshold adjusting for cost-sensitive classification.

14. Turney, P.: Types of cost in inductive concept learning. In Proc. of ICML'2000 Workshop on Cost-Sensitive Learning. Stanford, CA (2000).

15. Breiman, L., Friedman, R.A., Olshen, R., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984).

16. Bradford, J.P., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.E.: Pruning decision trees with misclassi_cation costs. In Proc. of the Tenth European Conf. on Machine Learning. Springer Verlag (1998) 131-136.

17. Knoll, U., Nakhaeizadeh, G., Tausend, B.: Cost-sensitive pruning of decision trees. In Proc. of the 8th European Conf. on Machine Learning. Springer LNCS 784 (1994) 383-386.

18. M. Pazzani C. Merz P. Murphy K. Ali T. Hume and C. Brunk, “Reducing Misclassification Costs, Proc. 11th Int'l Conf. Machine Learning, pp. 217-225, 1994.

19. Ting, K.M.: Inducing cost-sensitive trees via instance weighting. In Principles of Data Mining and Knowledge Discovery. 2nd European Symposium PKDD'98. Springer LNCS 1510 (1998) 139-147.