An Empirical Study of MetaCost Using Boosting Algorithms

Kai Ming Ting

School of Computing and Mathematics, Deakin University,
Vic 3168, Australia
kmting@deakin.edu.au

Abstract. MetaCost is a recently proposed procedure that converts an error-based learning algorithm into a cost-sensitive algorithm. This paper investigates two important issues centered on the procedure which were ignored in the paper proposing MetaCost. First, no comparison was made between MetaCost’s final model and the internal cost-sensitive classifier on which MetaCost depends. It is credible that the internal cost-sensitive classifier may outperform the final model without the additional computation required to derive the final model. Second, MetaCost assumes its internal cost-sensitive classifier is obtained by applying a minimum expected cost criterion. It is unclear whether violation of the assumption has an impact on MetaCost’s performance. We study these issues using two boosting procedures, and compare with the performance of the original form of MetaCost which employs bagging.

1 Introduction

MetaCost (Domingos, 1999) is a recently proposed method for making an arbitrary classifier cost-sensitive. The method has an interesting design that uses a “meta-learning” procedure to relabel the classes of the training examples, and then employs the modified training set to produce a final model. MetaCost has been shown to outperform two forms of stratification which change the frequency of classes in the training set in proportion to their cost.

MetaCost depends on an internal cost-sensitive classifier in order to relabel classes of training examples. But the study by Domingos (1999) made no comparison between MetaCost’s final model and the internal cost-sensitive classifier on which MetaCost depends. This comparison is worth making as it is credible that the internal cost-sensitive classifier may outperform the final model without the additional computation required to derive the final model.

A simple method of converting an error-based classifier to a cost-sensitive classifier is to apply an additional minimum expected cost criterion (Michie, Spiegelhalter & Taylor, 1994). Because this criterion only needs to be applied during classification, this method has the advantage of re-using the same learned model when the cost changes. MetaCost assumes its internal cost-sensitive classifier is obtained this way, and thus inherits the advantage. However, previous study (Ting & Zheng, 1998) has shown that this simple approach is not the

best method to minimize cost. It is therefore unclear whether the performance of MetaCost will be affected if the assumption is violated.

Boosting (Quinlan, 1996; Schapire, Freund, Bartlett, & Lee, 1997) can be effective and can be better than bagging (Bauer & Kohavi, 1999) in minimizing errors. MetaCost (Domingos, 1999) uses bagging internally. Using a boosting procedure in MetaCost may improve MetaCost’s performance. This is the reason why we choose to use boosting procedures in MetaCost in this paper.

This paper has two aims. First, to make a direct comparison between MetaCost and the internal cost-sensitive classifier on which MetaCost relies. Second, to investigate whether a violation of MetaCost’s assumption has an impact on its performance. In the next section, we describe MetaCost and boosting. In Section 3, we present the empirical comparison in four separate subsections. A discussion of the results is given in Section 4, followed by conclusions in the last section.

2 MetaCost and Boosting Procedures

2.1 MetaCost

MetaCost (Domingos, 1999) is based on the Bayes optimal prediction that minimizes the expected cost $R(j|x)$ (Michie, Spiegelhalter & Taylor, 1994):

$$R(j|x) = \sum_i^I P(i|x) \text{cost}(i, j),$$

where $P(i|x)$ is the probability of class i given example x and $\text{cost}(i, j)$ is the cost of misclassifying a class i example as class j .

The Bayes optimal prediction rule implies a partition of the example space into I regions, such that class i is the minimum expected cost prediction in region i . If misclassifying class i becomes more expensive relative to misclassifying others, then parts of the previously non-class i regions shall be re-assigned as region i since it is now the minimum expected cost prediction.

The MetaCost procedure estimates class probabilities using bagging (Bauer & Kohavi, 1999), and relabels the training examples with their minimum expected cost classes, and finally relearns a model using the modified training set.

We interpret the process of estimating class probabilities and applying the Bayes optimal prediction rule as constructing an internal cost-sensitive classifier for MetaCost. With this interpretation, we formalize the MetaCost procedure as a three-step process depicted in Figure 1.

The procedure begins to learn an internal cost-sensitive model by applying a cost-sensitive procedure which employs a base learning algorithm. Then, for each of the training examples, assign the predicted class of the internal cost-sensitive model to be the class of the training example. Finally, learn the final model by applying the same base learning algorithm to the modified training set.

Given $\mathcal{T}$: a training set containing N examples (x_n, y_n) where x_n is a vector of attribute-values and y_n is the class label, L : a base learning algorithm, $\mathcal{C}$: a cost-sensitive learning procedure, and $cost$: a cost matrix.

MetaCost($\mathcal{T}, L, \mathcal{C}, cost$)

(i) Learn an internal cost-sensitive model by applying $\mathcal{C}$:

$$H^* = \mathcal{C}(\mathcal{T}, L, cost).$$

(ii) Modify the class of each example in $\mathcal{T}$ from the prediction of H^* :

$$y_n = H^*(x_n).$$

(iii) Construct a model H by applying L to $\mathcal{T}$.

Output: H .

Fig. 1. The MetaCost procedure

The performance of MetaCost relies on the internal cost-sensitive procedure in the first step. Getting a good performing internal model in the first step is crucial to getting a good performing final model in the third step.

MetaCost in its original form (Domingos, 1999) assumes that its internal cost-sensitive procedure is obtained by applying the Bayes optimal prediction rule to an existing error-based procedure. Thus, the cost-sensitive procedure $\mathcal{C}$ consists of first getting the class probability from a model h defined as follows. Choose an error-based procedure $\mathcal{E}$ which employs a training set $\mathcal{T}$ and a base learning algorithm L to induce the model $h = \mathcal{E}(\mathcal{T}, L)$, without reference to cost. Given a new example x , h produces a class probability for each class:

$$P(i|x) = h(x), \text{ for each class } i.$$

Then, apply the Bayes rule or minimum expected cost criterion:

$$H^*(x) = \arg \min_j \sum_i P(i|x) cost(i, j).$$

However, a cost-sensitive procedure, that takes cost directly into consideration in the training process, is another option. In this paper, both types of cost-sensitive procedures are used to evaluate their effects on MetaCost. We use the error-minimizing boosting algorithm AdaBoost in place of $\mathcal{E}$, and a cost-sensitive version of the boosting procedure in place of $\mathcal{C}$. AdaBoost is often effective in minimizing errors. Both procedures are described in the next section.

2.2 Boosting Procedures

AdaBoost induces multiple individual classifiers in sequential trials, and a weight is assigned to each training example. At the end of each trial, the vector of

Given $\mathcal{T}$: a training set containing N examples (x_n, y_n) , L : a base learning algorithm, and $cost$: a cost matrix.

AdaBoost($\mathcal{T}, L, cost, K$)

Initialization: all instance weights $w_1(n) = 1$.

For $k = 1, \dots, K$

- (i) Learn a model h_k by applying L to $\mathcal{T}$ under the weight distribution w_k . Let $h_k(x)$ denotes the predicted class, and $\hat{h}_k^i(x) \in [0, 1]$ denotes the confidence level of the prediction for class i .
- (ii) $\mathcal{T}$ is classified using h_k . The error of this model, ϵ_k , is defined as

$$\epsilon_k = \frac{1}{N} \sum_{x_n \in \mathcal{T}: h_k(x_n) \neq y_n} w_k(n). \quad (1)$$

If $\epsilon_k \geq 0.5$ or $\epsilon_k = 0$, then all $w_k(n)$ is reset using bootstrap sampling, i.e., $w_k(n)$ is set zero and incremented one unit every time instance n is selected in the sampling with replacement process to select N samples, and the process continues from step (i).

- (iii) The instance weight $w_{(k+1)}$ for the next trial is created from w_k as follows:

$$w_{(k+1)}(n) = \begin{cases} w_k(n)F_k & \text{if } h_k(x_n) \neq y_n \\ w_k(n)/F_k & \text{otherwise} \end{cases} \quad (2)$$

where, $F_k = \sqrt{\frac{(1-\epsilon_k)}{\epsilon_k}}$.

Output:

$$P(i|x) \propto \sum_k \log(F_k) \hat{h}_k^i(x). \quad (3)$$

$$H^*(x) = \arg \min_j \sum_i \sum_k \log(F_k) \hat{h}_k^i(x) cost(i, j). \quad (4)$$

Fig. 2. The AdaBoost procedure that uses the minimum expected cost criterion

weights is adjusted to reflect the importance of each training example for the next induction trial. This adjustment effectively increases the weights of misclassified examples and decreases the weights of the correctly classified examples. These weights cause the learner to concentrate on different examples in each trial and so lead to different classifiers. Finally, the individual classifiers are combined to form a composite classifier. The AdaBoost procedure is shown Figure 2. Note that the weight adjustment formula in Equation (2) is from a version of AdaBoost described in Schapire, Freund, Bartlett, & Lee (1997). To use the simple method to convert the error-based AdaBoost procedure to a cost-sensitive version, a minimum expected cost criterion is applied as shown in Equation (4). This is made possible by assuming the weighted votes for each class is proportional to the class probability, as shown in Equation (3).

Initialization: The weight of every class j instance at $k = 1$ is initialized as follows.

$$w_1(n) = w^j = C^j \frac{N}{\sum_i C^i N^i},$$

where $C^i = \sum_j \text{cost}(i, j)$, is the cost of misclassifying a class i instance, and N^i is the number of class i instances; $N/\sum_i C^i N^i$ is the normalizing term such that $\sum_j w^j C^j = N$

The weight update rule:

$$w_{(k+1)}(n) = \begin{cases} w_k(n) \text{cost}(y_n, h_k(x_n)) & \text{if } h_k(x_n) \neq y_n \\ w_k(n) & \text{otherwise} \end{cases}$$

$w_{(k+1)}(n)$ needs to be normalized so that the sum of all $w_{(k+1)}(n)$ equals to N .

Fig. 3. Weight initialization and weight update rule for cost-sensitive boosting CSB

A cost-sensitive boosting procedure that takes cost directly into consideration in the training process can be obtained by replacing the weight update rule of Equation (2) in the AdaBoost procedure to a new rule depicted in Figure 3. The rule increases the weight by a factor of the misclassification cost if an example is misclassified, otherwise the weight from the previous trial is retained. Note that this is an improved version of the one proposed by Ting & Zheng (1998). We called this cost-sensitive boosting procedure, CSB. Although the changes also include a weight initialization process, also shown in Figure 3, the weight update rule has the dominating influence on the performance of CSB.

We denote MetaCost_A as the algorithm that uses AdaBoost in MetaCost, and MetaCost_CSB uses CSB. The base learning algorithm, L , we used to conduct our experiments is the well-known decision tree learning algorithm, C4.5 (Quinlan, 1993). Only the default settings of C4.5 are used. The parameter K controlling the number of classifiers generated in both boosting procedures is set at 10 for all experiments described in the following section, unless stated otherwise. When bagging is used, we also employ 10 classifiers in each run.

3 Experiments

In this section, we empirically evaluate the performance of MetaCost's final model H and its internal classifier H^* produced by boosting and bagging. Twenty-four natural datasets, which consists of fourteen two-class datasets and ten multi-class datasets, from the UCI machine learning repository (Blake, Keogh & Merz, 1998) are used in the experiments. The description of this test suite is shown in Table 1.

Table 1. Description of datasets

Dataset	Size	No. of Classes	No. of Attributes	
			Numeric	Nominal
breast cancer (Wisconsin)	699	2	9	0
liver disorder	345	2	6	0
credit screening (Australian)	690	2	6	9
echocardiogram	131	2	6	1
solar flare	1389	2	0	10
heart disease (Cleveland)	303	2	13	0
hepatitis prognosis	155	2	6	13
horse colic	368	2	7	15
house-voting 84	435	2	0	16
hypothyroid diagnosis	3168	2	7	18
king-rook-vs-king-pawn	3169	2	0	36
pima indians diabetes	768	2	8	0
sonar classification	208	2	60	0
tic-tac-toe end game	958	2	0	9
abalone	4177	3	7	1
annealing process	898	6	6	32
glass identification	214	6	9	0
lymphography	148	4	0	18
nettalk-stress	5438	5	0	7
nursery	12960	5	0	8
satellite	6435	6	36	0
soybean large	683	19	0	35
splice junction	3177	3	0	60
wine recognition	178	3	13	0

For each of the two-class datasets, we report the sum of three averages of two 10-fold cross-validations using three fixed cost ratios. Costs are assigned such that misclassifying a minority class example costs more than misclassifying a majority class example. Suppose i is the majority class, that is, $P(i) > P(j)$, then $cost(i, j) = 1$ and $cost(j, i) = r$. The fixed cost ratios used to obtain the three averages are $r = 2, 5$, and 10 . This means that misclassifying a minority class example is r times more costly than misclassifying a majority class example. In this way, we simulate the situation often found in practice where it is most important to correctly classify the rare classes. In this paper, all correct classifications are assumed to have no cost, that is, for all i , $cost(i, i) = 0$.

For each of the multi-class datasets, we report the average of two 10-fold cross-validations. We emulate the similar condition as in the two-class datasets. Each cost matrix is randomly initialized at the beginning of each run as follows. For each pair of i, j and $i \neq j$ and $P(i) > P(j)$, we assign $cost(i, j) = 1$ and $cost(j, i) = r$, where r is a randomly selected integer from 2 to 10.

We use two measures to evaluate the performance of the algorithms employed for cost-sensitive classification. The first measure is the *total cost of misclassifications* made by a classifier on a test set (i.e., $\sum_m cost(actual(m), predicted(m))$). The second measure is the *number of high cost errors*. It is the number of misclassifications associated with cost higher than 1 made by a classifier on a test

Table 2. Comparison of MetaCost, AdaBoost and CSB (Cost)

Dataset	MetaC_A	AdaB vs MetaC_A	MetaC_CSB	CSB vs MetaC_CSB	CSB vs AdaB	MetaC_CSB vs MetaC_A
	cost	cost ratio	cost	cost ratio	cost ratio	cost ratio
breast cancer	18.50	.85	20.35	.74	.96	1.10
liver disorder	57.75	.99	57.90	.99	1.00	1.00
credit	64.25	1.00	61.10	.97	.92	.95
echocardio.	23.15	.97	24.10	1.01	1.08	1.04
solar flare	215.15	1.02	226.05	1.02	1.05	1.05
heart (C)	39.30	.96	39.70	.95	.99	1.01
hepatitis	25.15	.90	21.95	.95	.91	.87
horse colic	53.40	.95	50.65	.99	.99	.95
house-voting	13.45	1.14	11.30	1.02	.75	.84
hypothyroid	27.60	1.11	25.45	1.01	.84	.92
kr-vs-kp	54.45	1.15	21.95	.86	.30	.40
pima	110.70	.99	112.35	.96	.98	1.01
sonar	26.40	.99	32.25	.76	.94	1.22
tic-tac-toe	114.95	.73	120.05	.68	.97	1.04
Mean Geomean	60.30	.98	58.94	.91	.87	.93
abalone	238.05	.99	231.90	1.01	.99	.97
anneal	18.45	.82	14.45	.71	.68	.78
glass	16.05	.85	15.05	.84	.93	.94
lymphography	8.40	.77	8.85	.76	1.03	1.05
nettalk-stress	257.75	.78	252.95	.79	.99	.98
nursery	149.65	.90	80.00	.86	.51	.53
satellite	201.65	.86	218.10	.72	.90	1.08
soybean	18.85	.72	13.70	.68	.69	.73
splice junction	74.10	.81	38.10	1.01	.64	.51
wine	2.65	1.70	3.60	.72	.58	1.36
Mean Geomean	98.56	0.89	87.67	.80	.80	.86
w/t/l p of w/l	18/1/5 .005		18/0/6 .0113		20/1/3 .0002	13/1/10 .3388

set. Both measures are presented because a classifier that minimizes the cost does not necessarily minimize the number of high cost errors. Depending on the need of the user, a good cost-sensitive classifier should have either low total misclassification cost or small number of high cost errors.

3.1 Total Misclassification Cost

Table 2 shows the results of the comparison between MetaCost_A and AdaBoost, and between MetaCost_CSB and CSB in terms of misclassification costs and cost ratios. A cost ratio for A vs B is a ratio of costs due to A and to B. A ratio of less than 1 for AdaBoost vs MetaCost_A, for example, represents an improvement

due to AdaBoost. The relative performance between CSB and AdaBoost, and between the two versions of MetaCost are shown in the last two columns. The first fourteen datasets are two-class datasets. A summary of the mean costs and geometric mean ratios over the fourteen two-class datasets are also shown. A similar summary is provided for the ten multi-class datasets in the second half of the table. A count of wins/ties/losses over the total twenty-four datasets, and the result of one-tailed sign tests on win/loss records are shown in the last two rows.

In terms of cost, we have the following observations:

- MetaCost usually does not perform better than its internal classifier. AdaBoost performs better than MetaCost_A in eighteen datasets, performs worse in five datasets, and has equal performance in one dataset. CSB performs better than MetaCost_CSB in eighteen datasets and worse in six datasets. Both cases are significant at a level better than 98%. MetaCost retains only a portion of performance of its internal classifier. Using CSB, which is a better performing cost-sensitive classifier than AdaBoost, MetaCost_CSB retains between 68% and 99% of CSB's performance. In only six out of twenty-four datasets, MetaCost_CSB performs comparably to CSB, with a maximum gain of 2% in relative cost.
- MetaCost performs better in two-class datasets than in multi-class datasets. On average, MetaCost_A retains 98% of AdaBoost's performance in two-class datasets, in comparison to only 89% in multi-class datasets. Similarly, MetaCost_CSB retains an average of 91% of CSB's performance in two-class datasets, in comparison to only 80% in multi-class datasets. This is because it is easier to concentrate on minimizing the total cost of problems with two classes, one high cost and one low cost, than that with multiple classes. In addition, an ensemble of multiple models, used in AdaBoost and CSB, is more readily adjustable to the multi-class multi-cost problems than a single model produced by MetaCost.
- Although CSB is significantly better than AdaBoost, MetaCost_CSB is not significantly better than MetaCost_A with thirteen wins and ten losses. Nevertheless, MetaCost_CSB performs better than MetaCost_A on average across the twenty-four datasets. In cases where MetaCost_CSB performs better, the gain can be as much as 60% in relative cost as in the kr-vs-kp dataset, or as much as 69 unit cost as in the nursery dataset. In cases where MetaCost_CSB performs worse, the maximum loss is only 16 unit cost in the satellite dataset. Thus, a better performing internal classifier does give MetaCost a better chance of producing a better performing final model.

3.2 Number of High Cost Errors

Table 3 shows the results of the same comparison in terms of the number of high cost errors. We have the following observations:

- MetaCost almost always performs worse than its internal classifier. MetaCost produces an average of over 30% more relative errors in two-class datasets

Table 3. Comparison of MetaCost, AdaBoost and CSB (High cost errors)

Dataset	MetaC_A	AdaB vs MetaC_A	MetaC_CSB	CSB vs MetaC_CSB	CSB vs AdaB	MetaC_CSB vs MetaC_A
	#hce	#hce ratio	#hce	#hce ratio	#hce ratio	#hce ratio
breast cancer	1.40	.43	2.30	.43	1.67	1.64
liver disorders	3.80	.86	4.90	.80	1.20	1.29
credit	4.35	1.11	6.40	1.02	1.34	1.47
echocardio.	2.00	.75	2.25	.93	1.40	1.13
solar flare	30.10	.99	32.75	.92	1.01	1.09
heart (C)	3.60	.74	4.30	.79	1.28	1.19
hepatitis	3.05	.87	2.80	.75	.79	.92
horse colic	3.40	1.04	4.35	1.09	1.34	1.28
house-voting	0.80	1.00	1.05	.90	1.19	1.31
hypothyroid	2.70	.94	3.65	.92	1.31	1.35
kr-vs-kp	1.40	.82	2.40	.60	1.26	1.71
pima	9.20	.87	9.75	.81	.99	1.06
sonar	2.50	.28	4.55	.33	2.14	1.82
tic-tac-toe	8.00	.16	10.05	.19	1.56	1.26
Mean	5.45		6.54			
Geomean		.69		.69	1.29	1.30
abalone	9.60	.95	7.35	1.13	.91	.77
anneal	0.95	.42	1.15	.52	1.50	1.21
glass	1.35	.63	1.35	.70	1.12	1.00
lymphography	0.45	.44	0.60	.50	1.50	1.33
nettalk-stress	18.45	.67	19.30	.67	1.05	1.05
nursery	2.85	.63	3.90	.56	1.22	1.37
satellite	16.00	.57	23.25	.40	1.03	1.45
soybean	1.00	.40	1.30	.42	1.38	1.30
splice junction	1.85	.73	2.65	.83	1.63	1.43
wine	0.05	.00	0.35	.29	* > 1.00	7.00
Mean	5.26		6.12			
Geomean		.58		.56	†1.24	†1.19
w/t/1		21/1/2		21/0/3	3/0/21	2/1/21
p of w/1		.0000		.0001	.0001	.0000

* divide by zero. † computed from the first nine figures of the multi-class datasets.

than its internal classifier, and over 40% in multi-class datasets. With twenty-one wins out of twenty-four datasets, the differences are significant at a level of 99.99% no matter the internal classifier is CSB or AdaBoost.

- As for the cost measure, MetaCost performs better in two-class datasets than in multi-class datasets. In two-class datasets, MetaCost retains an average of about 70% of the performance of its internal classifier. In multi-class datasets, MetaCost retains an average of about 60% of the performance of its internal classifier.
- A different from the cost-based measure is that MetaCost_A is significantly better in terms of the number of high cost errors than MetaCost_CSB at a level of 99.99%. MetaCost_A achieves a relative error of 30% improvement

over MetaCost_CSB in two-class datasets, and a relative error of more than 19% improvement in multi-class datasets. AdaBoost achieves a similar level of improvement over CSB. This shows that the performance of MetaCost improves in terms of the number of high cost errors if a better performing internal classifier is used.

3.3 The Effect of K

It has been shown that increasing K , the number of classifiers, in the boosting procedure can reduce the number of errors. It is interesting to see the effect of K on MetaCost and its internal boosting procedures in terms of cost and high cost errors.

Figure 4 shows an example of performance comparison between MetaCost_CSB and CSB as K in the boosting procedure increases from 5, 10, 20, 50, 75 to 100 classifiers in the satellite dataset. In terms of high cost errors, both MetaCost_CSB and CSB initially reduce the errors as K increases and then stabilize. Although CSB stabilizes earlier at $K = 20$, with comparison to MetaCost_CSB which stabilizes at $K = 75$, CSB always has fewer errors than MetaCost_CSB. Both MetaCost_CSB and CSB have similar profiles in the figures. As K increases cost initially falls, but then increases. For MetaCost_CSB, the cost increases beyond the point at which the high cost errors stabilized at $K = 75$; for CSB it is at $K = 20$. The increased total cost is due to the increase in low cost errors while the boosting procedure continues its effort to reduce high cost errors, eventually without success.

In terms of tree size, MetaCost_CSB produces a smaller tree as K increases, from a size of 550 nodes at $K = 5$ to 398 at $K = 100$. On the other hand, CSB produces a combined tree size of 2166 nodes at $K = 5$, and increases to 18881 at $K = 100$.

Fig. 4. Satellite: Comparing MetaCost_CSB with CSB in terms of cost and the number of high cost errors as K in the boosting procedure increases

3.4 MetaCost Using Bagging

MetaCost originally uses bagging as its internal classifier (Domingo, 1999). The aims of this section are to investigate how well it performs compared with MetaCost using boosting, and whether MetaCost’s final model performs better than bagging. Table 4 shows the result summary across twenty-four datasets. Detailed results can be found in Ting (2000).

Table 4. Result summary for MetaCost using Bagging, AdaBoost and CSB

	Bagging vs MetaC_B	Bagging vs AdaB	MetaC_B vs MetaC_A	MetaC_B vs MetaC_CSB
Cost ratio geomean	.82	.88	.96	1.06
w/t/l	20/1/3	14/1/9	11/0/13	18/0/6
p of w/l	.0002	.2024	.4194	.0113
#hce ratio geomean	.61	1.50	1.73	1.28
w/t/l	23/1/0	2/0/22	0/0/24	18/3/3
p of w/l	.0000	.0000	.0000	.0007

MetaCost using bagging is found to perform significantly worse than bagging, both in terms of cost and high cost errors. Bagging performs better than AdaBoost in terms of cost and the result carries over to MetaCost, but the differences are not significant. On the other hand, the reverse is true in terms of high cost errors and the differences are significant at a level better than 99.99%. In comparison to that using CSB, MetaCost using bagging performs significantly worse both in terms of cost and high cost errors.

In addition, we compute the percentage of training examples in which the original class is altered to a different class in step (ii) of the MetaCost procedure. Bagging modified an average of 9% of training examples across the twenty-four datasets, and AdaBoost modified an average of 22%. The additional modifications directly contribute to the better performance of MetaCost using AdaBoost over that using bagging in terms of high cost errors.

4 Discussion

The results in this paper show that using a weaker internal classifier, in terms of cost, such as AdaBoost may mislead one to suggest that MetaCost’s final model performs comparably with its internal classifier in terms of cost, especially in two-class datasets. Our results suggest strongly that a better performing cost-sensitive classifier should be used with MetaCost.

MetaCost’s assumption—of using an error-based procedure and then applying the minimum expected cost criterion—has an advantage of re-using the error-based model whenever the cost changes. Unfortunately, the best performing cost-sensitive classifier cannot be obtained this way, and require cost to be taken into consideration in the training process. Our results and the results of previous study (Ting & Zheng, 1998), though using different cost assignment for cost matrix, show that cost-sensitive boosting performs better than AdaBoost in

terms of cost. Further, the results carry over to MetaCost. Even in terms of high cost errors, where AdaBoost is significantly better than CSB, the performance of MetaCost_A can be further improved by taking cost directly into consideration in the training process by changing Equation (2) to

$$w_{(k+1)}(n) = \begin{cases} w_k(n)F_k \text{cost}(y_n, h_k(x_n)) & \text{if } h_k(x_n) \neq y_n \\ w_k(n)/F_k & \text{otherwise.} \end{cases}$$

This improved version of MetaCost_A gains a relative high cost error of 9% against MetaCost_A in both the two-class and multi-class datasets. The difference is significant at a level better than 95% with 17 wins and 7 losses.

Boosting has been shown to outperform bagging (Bauer & Kohavi, 1999) in reducing total errors. Our experiments in Section 3.4 clearly shows that the result does carry over to cost-sensitive classification and MetaCost in terms of reducing high cost errors. However, this result does not necessarily imply that AdaBoost is a better probability estimator than bagging. This is because poor probability estimates can still lead to optimal classification, as long as the class that minimizes expected cost (given the estimated probabilities) is the same as that which minimizes cost given the true probabilities (Domingos, 1999).

5 Conclusions

This paper has investigated two important issues centered on the MetaCost procedure which were ignored in the previous study. First, we find that MetaCost retains only part of the performance of the internal classifier on which it relies, both in terms of cost and high cost errors. This occurs for both boosting and bagging, as its internal classifier. We also find that the better performance of the internal classifier, the better is the chance of improving the final model for MetaCost. Second, using an internal cost-sensitive procedure, MetaCost is expected to perform better than with the original version involving an error-based procedure. This follows as a cost-sensitive procedure usually performs better than an error-based procedure for cost-sensitive classification.

Based on our results, we do not recommend using MetaCost when the aim is to minimize the misclassification cost or the number of high cost errors. MetaCost is only recommended if the aim is to have a more comprehensible model and the user is willing to sacrifice part of the performance.

Acknowledgement

The anonymous reviewers, Geoff Webb and Russell Rimmer provided helpful suggestions to improve the paper. Ardena assisted with the experiments. This research is supported in part by a Deakin University Seeding Grant.

References

1. Bauer, E. & Kohavi, R. (1999), An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36, pp. 105-139, Kluwer Academic Publishers.
2. Blake, C., Keogh, E. & Merz, C.J. (1998), *UCI Repository of machine learning databases* [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. Irvine, CA: University of California, Dept. of Information and Computer Science.
3. Domingos, P. (1999), MetaCost: A General Method for Making Classifiers Cost-Sensitive, in *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, ACM Press.
4. Michie, D., D.J. Spiegelhalter, & C.C. Taylor (1994), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Limited.
5. Quinlan, J.R. (1993), *C4.5: Program for Machine Learning*, Morgan Kaufmann.
6. Quinlan, J.R. (1996), Bagging, boosting, and C4.5, in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 725-730, AAAI Press.
7. Schapire, R.E., Y. Freund, P. Bartlett, & W.S. Lee (1997), Boosting the margin: A new explanation for the effectiveness of voting methods, in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 322-330.
8. Ting, K.M. (2000), An Empirical Study of MetaCost using Boosting Algorithms. *Technical Report C01/00*, Deakin University.
9. Ting, K.M. & Z. Zheng (1998), Boosting Cost-Sensitive Trees, *Proceedings of the First International Conference on Discovery Science*, LNAI-1532, Berlin: Springer-Verlag, pp. 244-255.