Improving Classifier Utility by Altering the Misclassification Cost Ratio

Michelle Ciraco, Michael Rogalewski and Gary Weiss

Department of Computer Science
Fordham University
Rose Hill Campus
Bronx, New York 10458

{ciraco, rogalews, gweiss}@cis.fordham.edu

ABSTRACT

This paper examines whether classifier utility can be improved by altering the misclassification cost ratio (the ratio of false positive misclassification costs to false negative misclassification costs) associated with two-class datasets. This is evaluated by varying the cost ratio passed into two cost-sensitive learners and then evaluating the results using the actual (or presumed actual) cost information. Our results indicate that a cost ratio other than the true ratio often maximizes classifier utility. Furthermore, by using a hold out set to identify the "best" cost ratio for learning, we are able to take advantage of this behavior and generate classifiers that outperform the accepted strategy of always using the actual cost information during the learning phase.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning – *Induction*
H.2.8 [Database Management]: Applications - *Data Mining*

General Terms

Algorithms

Keywords

Data mining, machine learning, induction, cost-sensitive learning, utility-based data mining

1. INTRODUCTION

Classifier induction programs have traditionally made a number of assumptions. One such assumption is that the best class distribution for learning is the true underlying distribution—the one that the classifier will eventually be applied to. Recent research has shown that this assumption is often not true and that improved performance can be achieved by using a modified class distribution [6]. This leads us to ask a similar question, "can classifier performance be enhanced by employing cost-sensitive learning and altering the ratio of true misclassification costs?" That is, can we obtain better classifier performance by training with a cost ratio that is not the same as the one that will be used to evaluate the classifier? This is the principle question that we investigate in this paper and, somewhat to our surprise, the answer is "yes".

Much early work on machine learning and data mining did not consider issues of utility, including how a classifier will be used. In particular, misclassification errors typically have non-uniform costs. These misclassification costs are often determined by the class associated with an example, such that for two-class problems, the cost of a false positive prediction is not equal to the cost of a false negative prediction. In these circumstances accuracy is a poor utility metric [4] and for that reason we consider cost information when evaluating the utility of a classifier.

In this paper we vary the cost ratios employed in the learning phase, and analyze how this impacts the performance of the classifier, based on the presumed "actual" cost information associated with the data set. Besides helping us answer the practical question posed earlier, about whether one can improve performance by altering the cost ratio during learning, a secondary benefit is that we gain insight into cost-sensitive learning.

2. BACKGROUND AND TERMINOLOGY

We begin by introducing the basic terminology used to describe the behavior of a classifier for a two-class data set. This is provided in Table 1, which shows a confusion matrix. Note that in Table 1 "ACTUAL" represents the true class for an example, whereas "PREDICTED" represents the label assigned by the classifier. Misclassified examples are those that are either false positives or false negatives and accuracy is defined as: (TP + TN)/(TP + FP + TN + FN).

Table 1: Confusion Matrix Terminology

	ACTUAL		
PREDICTED		Positive class	Negative class
	Positive class	True positive (TP)	False positive (FP)
	Negative class	False negative (FN)	True negative (TN)

One can apply different cost (or profit) values to each of the four possible outcomes listed in Table 1. For cost-sensitive learning, one typically specifies only the costs for the false positives and false negatives and assigns a cost of zero to the true positives and

UBDM '05 , August 21, 2005, Chicago, Illinois, USA.

true negatives. In this case, the construction of the classifier will only be affected by the ratio of the two non-zero misclassification costs. Throughout this paper we specify the cost ratios as the ratio of false positive costs to false negative costs. So, a cost ratio of 1:5 indicates that the cost of a false negative is five times that of a false positive. Holding with established convention, the minority class is considered the positive class. This is often the class of primary interest, as in the case of medical diagnosis (most patients do not have the sickness for which they are being tested). In these situations, the cost assigned to a false negative is generally greater than the cost assigned to a false positive, since this leads to classifiers that correctly classify a greater percentage of the minority-class examples.

In this paper, we are interested in two cost ratios. The actual cost ratio is based on the characteristics of the domain and is typically provided by a domain expert. We refer to this as the evaluation misclassification cost ratio, or, more simply, as the Evaluation Cost Ratio, abbreviated ECR. This name reflects the fact that this cost ratio is used when *evaluating* the utility of the classifier. In particular, we measure the quality of the classifier based on total cost, which is calculated using the equation below (the evaluation cost ratio is $FP_{cost}: FN_{cost}$).

$$\text{Total cost} = FP * FP_{cost} + FN * FN_{cost} \qquad [1]$$

Under normal circumstances, the evaluation cost ratio is passed to the classifier induction program, assuming that it is capable of cost-sensitive learning. However, in this paper we often utilize a different cost ratio in the learning phase. We refer to this ratio as the Training misclassification Cost Ratio, abbreviated TCR. For most of our experiments, TCR ≠ ECR. TCR affects the *construction* of the classifier but not the evaluation of the classifier.

Our paper is organized as follows. In Section 3 we describe our experiments. Results are then described in Section 4 and are discussed in Section 5. Section 6 describes related work and Section 7 presents our conclusions.

3. EXPERIMENTS

3.1 Data Sets

Our empirical study analyzes the relationship between TCR and ECR for the twelve data sets described in Table 2.

Table 2: Description of Data Sets

Data Set	Number of Attributes	Minority Class %	Total Size
Cover-Type*	55	48%	581,012
Adult	15	24%	21,281
Coding	16	50%	20,000
Letter-Vowel*	17	24%	16,122
Blackjack+	5	36%	15,000
Boa1+	69	50%	11,000
Mushroom	23	48%	8,124
Weather+	36	40%	5,597
Splice-Junction*	62	24%	3,175
Move+	11	50%	3029
OCR1	577	18%	2,283

The majority of the twelve data sets are available from the UCI Repository [3]. The remaining ones, identified by a "+" sign in Table 2, were originally supplied by AT&T and have been used in several published research papers. All of the data sets employed in this study are two-class data sets. Those that originally had more than two classes are identified by a "*". These were converted to two-class data sets by assigning one class, typically the least frequently occurring class, to the minority class, and then mapping all remaining classes to a single, majority, class.

3.2 Classifier Induction Programs

The majority of experiments described in this paper utilize C5.0, a commercial decision tree induction tool by Rulequest research. C5.0 is an updated, commercial version of Quinlan's popular C4.5 program [5]. To demonstrate the generality of our results, some experiments were repeated using the decision tree tool incorporated into Enterprise Miner™, an integrated suite of data mining tools from SAS Corporation. Both learners are capable of cost-sensitive learning.

3.3 Experimental Methodology

The primary purpose of the experiments described in this paper is to determine if one can improve classifier learning by using a training cost ratio (TCR) that is not equal to the evaluation cost ratio (ECR). Thus, the experiments in this paper vary the TCR value used during the learning phase and then evaluate the induced classifiers using the evaluation cost ratio. In order to run our experiments, we need to specify the TCR and ECR values for each data set. Because actual cost information is not available for most of the data sets we analyze (either because the costs are not known or not provided), we evaluate a wide range of ECR values rather than just one value. Because we are interested in how different TCR values impact performance, we also evaluate a wide variety of these values. For simplicity, we evaluate the same set of cost ratios for training (TCR) and evaluation (ECR). For each of the twelve datasets we evaluate the following nineteen cost ratios, for training and evaluation: 1:10, 1:9, … 1:2, 1:1, 2:1, …, 9:1, 10:1.

If we only wanted to understand the relationship between TCR and ECR (i.e., how a TCR value affects the performance of a classifier with a specified ECR), or whether a TCR equal to ECR yields the best classifier performance, then we would only need a training set for learning and a test set for classifier evaluation. However, we want to go further; we want to come up with a strategy for selecting, during the learning phase, a good TCR for learning, such that the utility of the induced classifier is improved. We therefore utilize a hold out set to identify the best TCR for learning (i.e., the one that yields the lowest total cost on the hold out set). We refer to this strategy as the TCR identification strategy, or simple TCR identification. The results using the TCR identification strategy are therefore based on using this identified TCR for classifier construction and then evaluating this classifier on the independent test set. Note, however, that since we are also interested in understanding how the choice of TCR and ECR impact classifier performance, we also report results using other TCR values. For the C5.0 experiments in this paper, each data set is partitioned as follows: 40% for the training set, 30% for the hold out set and 30% for the test set.

The above description was for our primary learner, C5.0. Our experiments that use the decision tree tool that is part of Enterprise Miner are slightly different. When using Enterprise Miner, we used the following seven cost ratios for training and evaluation:

1:10, 1:5, 1:2, 1:1, 2:1, 5:1, 10:1. We used fewer cost ratios for our Enterprise Miner experiments because of our time constraints and because it is much harder to automate the experiments when using Enterprise Miner (automation is more difficult because Enterprise Miner employs a graphical user interface, whereas C5.0 uses a command line interface). Furthermore, we did not utilize a hold out set for these experiments. For this reason, for our Enterprise Miner results we focus on how the choice of TCR affects total cost for different ECR values. Nonetheless, as we discuss later, the Enterprise Miner results tend to support the results and conclusions obtained using C5.0.

4. RESULTS

Experiments were run to determine the total costs produced by C5.0 and Enterprise Miner when the TCR and ECR values were varied for the data sets used in our study. In Section 4.1 we provide a detailed analysis of the coding data set. In Section 4.2 we briefly describe the detailed results for some other data sets and provide pointers to an on-line appendix [7] that includes the detailed statistics for all twelve data sets. Section 4.3 then provides the data set specific results using Enterprise Miner. This is followed by Section 4.4, which provides summary results for both C5.0 and Enterprise Miner. Our most important results are presented in Section 4.4.

4.1 Detailed Analysis of the Coding Data Set

We begin by showing the results for the Coding data set using C5.0. We chose to show detailed results of the coding data set because they are representative of the data sets that benefited from the TCR identification strategy. Table 3 shows the confusion matrix values for the Coding data set. Table 3 shows that as the cost ratio of FP to FN changes from 1:10 to 10:1 and false positive predictions become much more costly, then more negative predictions are made and the number of false positives decreases while the number of false negatives increases. We see that once the cost ratio reaches 3:1, every example is classified as a negative example.

Table 3: Confusion Matrix Values for Coding Data Set

TCR	FP	FN	TP	TN
1:10	2073	119	3381	427
1:9	1981	180	3320	519
1:8	1876	234	3266	624
1:7	1812	259	3241	688
1:6	1752	287	3213	748
1:5	1465	482	3018	1035
1:4	1303	634	2866	1197
1:3	1177	767	2733	1323
1:2	933	1104	2396	1567
1:1	592	1676	1824	1908
2:1	245	2502	998	2255
3:1	0	3500	0	2500
…	0	3500	0	2500
10:1	0	3500	0	2500

The utility of the classifier, as measured by total cost, is impacted by the evaluation cost ratio. Figure 4 shows the ECR curves for the coding data set, for ECR values 1:4, 1:3, 3:1 and 4:1 (for readability we do not show the ECR curves for all nineteen evaluated ECR values). Note that all of the curves flatten out at a TCR of 3:1, the point at which all examples are classified as the negative class. The different total cost values are due to the fact that three of the four curves have different false positive costs.

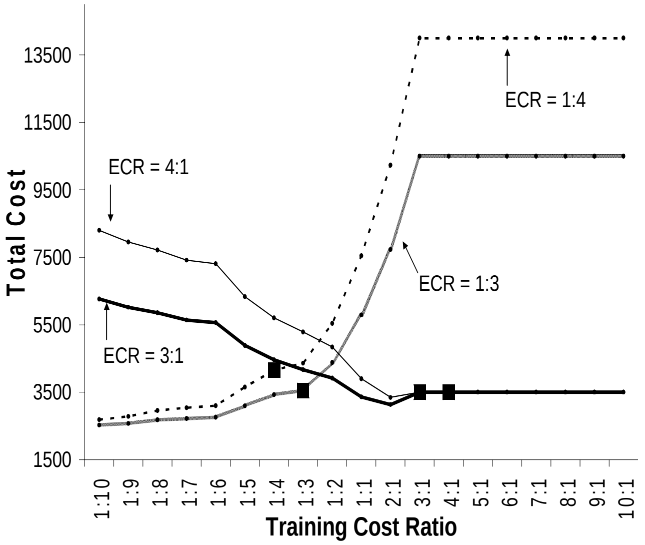

Figure 1: ECR Curves for the Coding Data Set

Each curve in Figure 1 has a square marker to indicate the point at which TCR=ECR. In none of the cases does this point yield the minimum total cost. This is especially true when the ECR is 1:4 or 1:3, in which case the savings appears to be very significant. This indicates that one might be able to improve classifier performance by selecting a TCR that is not equal to the ECR. However, for the TCR identification strategy to yield improved classifier performance, the results on the test data must be similar to the results for the hold out set, which are shown in Figure 1.

This leads us to Table 4, which compares the effectiveness of the TCR identification strategy to two other strategies for selecting the training cost ratio. The default strategy is what is commonly done—always setting the TCR to the ECR. The omniscient strategy involves selecting the TCR that produces the best results on the *test* set and then using that for learning. This strategy is not one that can be fairly used in practice, since it requires an oracle capable of knowing the performance of the classifier on future examples. However, it does provide an upper bound on the possible savings due to the TCR identification strategy and also allows us to see how effective the hold out set results are at identifying the optimal TCR for learning. For TCR identification and the omniscient strategy, the selected TCR value is specified along with the resulting total cost (for the default strategy the TCR is always equal to the ECR and so is not specified explicitly). The last two columns compare TCR identification to the default strategy and the omniscient strategy to the default strategy. The savings that are listed are relative savings. Note that all results in Table 4 are based on the test set. A TCR value of 2-10:1 is equivalent to 2:1-10:1 and means that all nine TCR values in this range yield identical results.

Table 4: Comparison of Three TCR Selection Strategies

ECR	Default	TCR Identification		Omniscient		% Savings (vs. Default)	
	Cost	TCR	Cost	TCR	Cost	TCR Identification	Omniscient
1:10	3664	1:10	3664	1:10	3664	0	0
1:9	3843	1:10	3501	1:10	3501	8.9	8.9
1:8	4052	1:10	3338	1:10	3338	17.6	17.6
1:7	3987	1:10	3175	1:10	3175	20.4	20.4
1:6	3782	1:10	3012	1:10	3012	20.4	20.4
1:5	4200	1:10	2849	1:10	2849	32.2	32.2
1:4	4153	1:10	2686	1:10	2686	35.3	35.3
1:3	3552	1:10	2523	1:10	2523	29.0	29.0
1:2	3229	1:6	2422	1:9	2359	25.0	26.9
1:1	2289	1:4	1972	1:3	1930	13.9	15.7
2:1	2926	1:1	2826	1:1	2826	3.4	3.4
3:1	3500	2:1	3136	2:1	3136	10.4	10.4
4:1	3500	2:1	3346	2:1	3346	4.4	4.4
5:1	3500	2-10:1	3500	2-10:1	3500	0	0
6:1	3500	2-10:1	3500	2-10:1	3500	0	0
7:1	3500	2-10:1	3500	2 -10:1	3500	0	0
8:1	3500	2-10:1	3500	2:-10:1	3500	0	0
9:1	3500	2-10:1	3500	2:-10:1	3500	0	0

The results in Table 4 demonstrate that in many cases very substantial reductions in cost are possible if TCR identification is used instead of the default strategy. The majority of situations where this is not true are when the TCR value is greater than 4:1. Note, however, that we are generally most concerned with the TCR values where the positive (minority-class) examples are given more weight—and this occurs in the range 1:2 to 1:10. Table 4 also shows us that the TCR identification strategy yields the same performance as the omniscient strategy in all but two cases (for ECR values of 1:2 and 1:1). The underlying data indicates that the hold out set identifies the TCR that gives the best test set results for all but these two cases.

4.2 Additional Detailed Results

In section 4.1 we provided the detailed results for the coding data set, using C5.0. In this section we briefly describe the results for the other eleven data sets included in our study, using C5.0. Due to space considerations, the detailed results are included in the on-line appendix [7], available at *http://storm.cis.fordham.edu/~gweiss/ubdm05-appendix.html*. However, summary statistics for all twelve data sets are provided in Section 4.3.

Figures analogous to Figure 1, but for the letter-vowel and adult data set, are provided in Figures A1 and A2 in the on-line Appendix, respectively. Tables analogous to Table 4 are also provided in the on-line appendix for all twelve data sets (Tables A1-A12). The one difference is that the tables in the appendix include one additional field, labeled "Hold Out Set Effectiveness". This field describes how close the hold out data set came to predicting the TCR with the lowest total cost. A value of '1' in that column means that the hold out set successfully predicted the TCR with the lowest total test-set cost. A value of *n* means that the hold out set predicted the TCR with the *n*th lowest total cost, when evaluated on the test data; thus a value of 19 would indicate that the TCR identification strategy identified the worst TCR value (since we evaluate 19 total TCR values). Based on Tables A1-A12, we see that the TCR identification strategy often identifies the TCR with the lowest total cost, and when it does not, it usually comes close.

If we analyze the behavior of the letter-vowel data set, using Figure A1 and Table A4, we see that TCR identification improves classifier performance over the default strategy in almost all cases, but unlike the results for the coding data set, the most substantial improvements are in the range 2:1-9:1, where the false positive cost is greater than the false negative cost. If we look at the results for the adult data set, in Figure A2 and Table A2, we see more ambiguous results. For that data set, TCR identification leads to small reductions in total cost in some cases and small increases in others. For detailed dataset-specific results, see Tables A1-A12 in the on-line appendix.

4.3 Enterprise Miner Results

Figures 2 and 3 show the ECR curves for the Adult data set and the Boa1 data set, respectively, when using SAS Enterprise Miner. As with the majority of C5.0 results, we see that the TCR that yields the best results is not always the one that equals the ECR. For Figure 2, the TCR value that generates the lowest total cost is always either 1:10 or 10:1; thus when the ECR value is not equal to one of these values (for an ECR of 1:5 or 5:1), suboptimal results are produced.

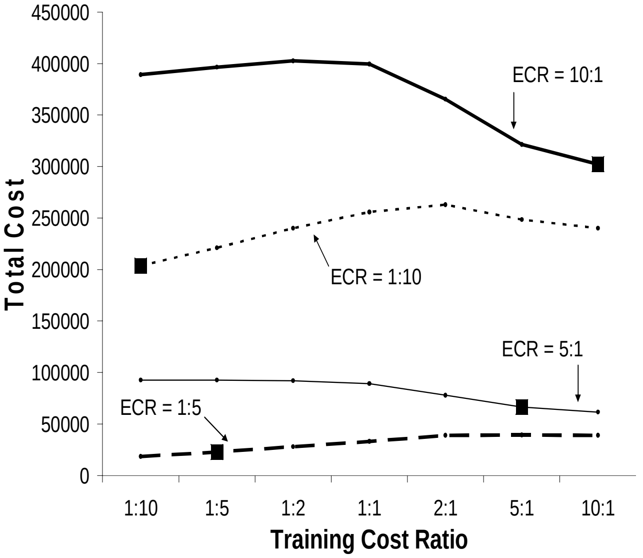

Figure 2: Enterprise Miner ECR Curves for Adult Data Set

Figure 3 shows that suboptimal results occur when the ECR is 1:2, since the best results occur for TCR values of 1:5 and 1:10, and for an ECR of 2:1 the best results occur for a TCR of 5:1 or 10:1. A detailed analysis of the Enterprise Miner results shows that, as with C5.0, the default strategy of setting the TCR equal to the ECR often does not provide the best performance. Section 4.4, which provides summary results for all C5.0 and Enterprise Miner data sets, will quantify the potential savings.

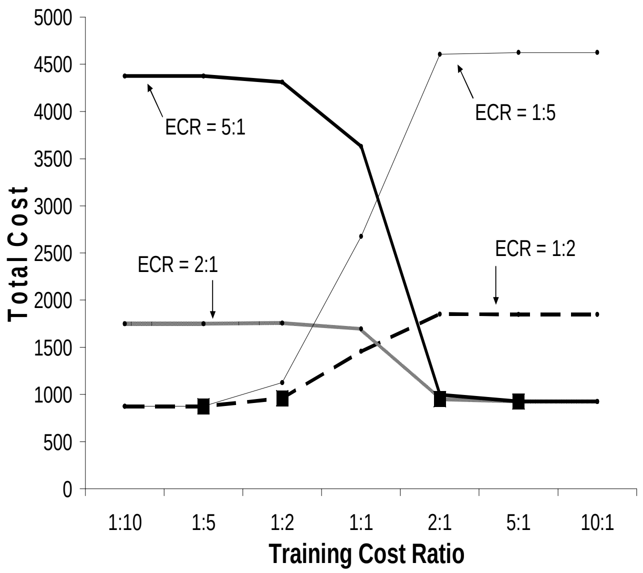

Figure 3: Enterprise Miner ECR Curves for Boa1 Data Set

4.4 Summarized Results over all Data Sets

This section summarizes the results over all of the data sets. The purpose of this section is to quantify the effectiveness of the TCR identification strategy and to identify any patterns or trends in the results. Table 5 shows how TCR identification compares to the default strategy of setting the TCR equal to the ECR, when evaluating classifier performance using total cost. The comparison is based on the performance over all nineteen cost ratio values.

Table 5: Comparison of TCR Identification vs. Default (C5.0)

Data Set	% Avg. Savings (1:10 – 10:1)	% Avg. Savings (1:10 -1:1)	Win/Loss/Tie
Cover-Type	-30.8% (-33.6%)	0.7%	4/13/2
Adult	0.0% (0.0%)	0.3%	6/8/5
Coding	11.6% (18.4%)	20.3%	12/0/7
Letter-Vowel	7.4% (8.3%)	1.4%	15/2/2
Blackjack	-0.2% (-1.1%)	-0.4%	1/3/15
Boa1	0.0% (0.0%)	0.0%	0/0/19
Mushroom	47.4% (100.0%)	60.0%	9/0/10
Weather	-0.2% (-0.4%)	0.3%	4/8/7
Network1	5.4% (7.3%)	5.7%	12/2/5
Splice-Junction	2.0% (2.2%)	-5.9%	8/9/2
Move	3.9% (9.3%)	3.4%	4/4/11
OCR1	23.4% (24.7%)	11.4%	15/3/1
TOTAL	5.8% (11.3%)	8.1%	90/52/105

The second column in Table 5 specifies the savings (i.e., relative reduction in cost) averaged over the full range of nineteen cost ratios. In parenthesis next to this number is the average savings if all ties between the TCR identification and default strategies are omitted. The third column shows the average savings over the ten cost ratios from 1:10 – 1:1. We break down the results for these cost ratios separately because, as mentioned earlier, one is typically most interested in this range since it leads to improved performance on the minority class, which otherwise might rarely be predicted. The last column in Table 5 provides the win/loss/tie record for each data set over the nineteen cost ratios.

Table 5 shows that the TCR identification performs substantially better than the default strategy of always using the ECR for training, although the performance varies widely for each data set. For the five data sets highlighted in bold and for which the row is shaded, TCR identification greatly outperforms the default strategy, in that it performs better for almost all of the nineteen cost ratios and consistently produces substantial savings. The four data sets that produced essentially neutral results are underlined. These include the boa1 data set, which produced perfectly identical results to the default strategy for all nineteen cost ratios, as well as the adult, blackjack and weather data set, which produced either very slight positive or negative savings. The move and splice-junction data sets can be considered moderate wins, when the entire range of cost ratios is considered. Finally, the Cover-type data set is the only true loss, over all nineteen cost ratios. However, for this data set the TCR identification strategy provides no loss (i.e., a very slight win) over the TCR range we are most interested in (1:10-1:1). If we focus exclusively on the range 1:10 – 1:1, then we see that there is only one moderate loss (splice-junction), while there are still many substantial wins.

In summary, our results indicate that of the twelve cases, averaged over the nineteen cost ratios, TCR identification produces substantially better results than the default strategy in five cases and worse results in one case; the remaining six cases produce ambiguous results. If we focus on the more important 1:10-1:1 TCR range, then the one significant loss is eliminated and the only loss at all is a moderate one, for the splice-junction data set. Due to space considerations, Table 5 does not compare the TCR identification strategy to the omniscient strategy, but that information is provided, at a more detailed level, in tables A1-A12 in the on-line appendix.

We now turn to the summary results for Enterprise Miner (EM). Because we did not utilize a hold out set for identifying the best TCR to use, we can only compare the omniscient strategy to the default strategy. This will put an upper bound on the savings that are possible. However, we expect that most of the savings we see with the omniscient strategy could be realized if a hold out set were used, since this is what we saw with C5.0. The results of this comparison are shown in Table 6.

Table 6: Comparison of Omniscient vs. Default Strategy (EM)

Data Set	% Avg. Savings
Cover-Type	22.7%
Adult	11.1%
Boa1	4.1%
Weather	19.1%
Network2	46.2%
Splice-Junction	1.9%
Move	18.7%
TOTAL	17.7%

Table 6 clearly demonstrates that the best TCR for learning is not equal to the ECR. While we have not demonstrated that these savings can be realized, we feel that the use of a hold out set would allow us to realize most of these savings.

5. DISCUSSION

This paper looks at two questions: does the evaluation cost ratio always produce the best results when used for training and 2) can we identify an alternative cost ratio for training such that classifier performance is improved. Our results indicate that the answer to the first question is "no", the best cost ratio for training is generally not the evaluation cost ratio and that the answer to the second question is "yes", we can identify an cost ratio for learning that outperforms the evaluation cost ratio.

Our main results show only one substantial failure for the TCR identification strategy, for the cover-type data set when all nineteen cost ratios were considered. Why were the results poor in this case? The detailed results for this data set, available in Table A1 in the on-line appendix, shows us the reason is that the hold out set did not effectively identify the best TCR, with respect to the test set performance, when the cost ratio was between 1:4 and 10:1. Even in the cases where our strategy did poorly, the omniscient strategy performed well. Our results may have been better for this data set, and for all data sets, had we employed multiple runs and averaged the results. In the only other failure, for the splice-junction data set for cost ratios 1:10 – 1:1, the failure resulted because the best TCR for learning, based on the test set, was 1:6, and the TCR identification strategy identified the best TCR (using the hold out data) as 1:7 (see Table A10). Thus, this failure was due to a slight difference between the selected and optimal TCR values.

Our results with Enterprise Miner support the conclusions reached using C5.0. Enterprise Miner also performed better when the TCR was set to a value other than the ECR. In the future we plan to incorporate the use of a hold out set to show that much of the potential reduction in total cost that is available can be realized.

Our study evaluated nineteen different evaluation cost ratios. An ECR of 1:1 is special, since it corresponds to the accuracy metric, which is very commonly used and therefore is, most likely, the metric for which most classifiers are optimized. Thus, it is worthwhile to analyze our results for an ECR of 1:1. For this ECR, for five of the twelve data sets a TCR of 1:1 provides the best performance and for three data sets the best performance is achieved with a TCR of 2:1. For the remaining four data sets, the best performance is achieved with a cost ratio of 1:3, 1:4, 3:1, or 4:1. No other ECR performed as well using the default strategy. For instance, with an ECR of 4:1, the best TCR ranged from 1:4 to 9:1 and a TCR of 4:1 yielded the lowest total cost only for one data set. Thus, our results indicate that the default strategy of setting the TCR to the ECR *is* generally quite appropriate for maximizing classifier accuracy.

This paper has focused on the issue of how the choice of the training cost ratio impacts classifier performance, as measured by total cost. Because one can effectively modify the misclassification cost ratio for a data set by altering the class distribution of the training set[1], our results have some additional implications. Specifically, our results seem to imply that one should often be able to benefit by altering the class distribution of the domain. In fact, this conclusion is seemingly supported by existing research [6], which showed that the naturally occurring class distribution often does not provide the best classifier performance. This connection, however, is not as clear. The research previously noted [6] showed that altering the class distribution improves learning when the training set size is held fixed; it did not examine the case where training examples were added. Also, the results in this paper can only be used to conclude that altering the class distribution will improve learning only if one can draw new training examples. However, most research on altering the class distribution of a training set utilizes sampling to change the distribution. Sampling involves either discarding examples of one class (undersampling) or duplicating examples of one class (oversampling). Both of these methods have drawbacks; undersampling throws away potentially useful data and oversampling may lead to overfitting the data. Thus, even though there is an equivalency between changing the cost ratio and altering the training sets class distribution, it is not clear that our results show that changing the class distribution of the training data will necessarily improve classifier performance.

Finally, we turn to the question of why the best cost ratio for training is not always the evaluation cost ratio. It is useful to start with accuracy, which is based on an ECR of 1:1. Our results indicate that in this case the default strategy of setting TCR to ECR performs quite well. This is encouraging, since it doesn't seem likely that a classifier would consistently perform sub-optimally for accuracy, the metric that most classifiers were originally optimized for. However, that still leaves us with the question as to why C5.0 and Enterprise Miner perform best for a training cost ratio other than the evaluation cost ratio when there are non-uniform misclassification costs. The only apparent answer is that these cost-sensitive learners do not handle non-uniform misclassification costs well. We are not sure why this is true, but feel that is an important area for future research. The only insight we have is that, for decision tree learning, it may be more difficult to label a leaf node with the correct class for non-uniform error costs, where the class probability threshold will not be 0.5. Our conjecture is that as the decision threshold moves away from 0.5, it becomes more difficult to accurately label the node, especially if there are only a small number of training examples. However, the evaluation of this conjecture is left for future work.

6. RELATED WORK

Research by Weiss and Provost [6] looked at the impact of class distribution on learning, when training data is costly and one is only able to purchase a fixed number of examples. That research found that improved classifier performance was possible by using a class distribution other than then naturally occurring distribution. That article then went on to show that one could use an adaptive, progressive, sampling strategy to identify a "good" class distribution for learning and thus actually improve classification performance. In many ways the research in this paper parallels that research, except that here we alter the cost ratio of the training set instead of its class distribution. In fact the results here might appear to be implied by those earlier results, since altering the class distribution of the training data is, as Elkan [2] pointed out, in some ways equivalent to altering the cost ratio. However, there is an important difference. In the earlier work, when the class distribution was changed, measures were taken to adjust the

[1] For example, to impose a misclassification cost ratio of 1:2 without using a cost-sensitive learner, one need only increase the ratio of positive to negative examples by a factor of 2 [2].

classifier so that it was not biased to favor the over-sampled class. Thus, changing the class distribution was not equivalent to altering the cost ratio, and the results in this paper are not implied by that earlier work.

Domingos [1] developed a method called Metacost that can transform a wide variety of error-based classifiers into cost-sensitive classifiers. Metacost re-labels the training set examples with their optimal class, or the class that minimizes conditional risk, and then relearns the classifiers with the modified training data. While both Metacost and the TCR identification method alter the training data or how the classifier treats the training data, the methods are incomparable because 1) TCR identification requires a cost-sensitive learner and 2) TCR identification identifies a cost ratio that is different from the one that would be derived from the cost matrix, while Metacost uses the one derived from the cost matrix (i.e., the ECR). However, Metacost could be adapted to do something similar to TCR identification and alter the class probability thresholds so that TCR ≠ ECR for the entire classifier, or could even use different class probability thresholds for different parts of the classifier (e.g., different rules).

There is great deal of additional research on cost-sensitive learning, but we are not aware of any studies that examine the impact of the changing the cost ratio during the learning process in order to assess the impact that this has on the quality of the induced classifier. This can be contrasted to the wealth of research that examines how changing the class distribution can improve learning from skewed class distributions.

7. CONCLUSION

In this paper we demonstrated that the performance of a classifier, when the misclassification costs are not uniform, is generally maximized when the training cost ratio is set equal to a value other than the evaluation cost ratio. This was shown for commercially available classifier induction programs, C5.0 and Enterprise Miner. We furthermore showed that, for C5.0, we could successfully identify good training cost ratios for learning, using a hold out test set, such that classifier performance could be improved over the standard practice of setting the training cost ratio to the evaluation cost ratio. Our results showed that for five of twelve data sets, the TCR identification strategy led to substantial reductions in total cost and only in one case did it lead to substantial increases in total cost. Furthermore, when the evaluation of the TCR evaluation strategy considered only cost ratios between 1:10 and 1:1, this one loss disappeared. The implications of our results are significant—that classifier induction programs may perform poorly—unnecessarily poorly—when handling non-uniform misclassification costs. These results are particularly notable since we analyzed popular state-of-the-art commercial classifiers. The "wrapper" approach we introduced in this paper can be used to overcome some of the weaknesses of these cost-sensitive learners.

The fact that the TCR identification strategy yields a net improvement in classifier performance indicates that the induced learners exhibit a systematic bias (altering the cost ratio will always increase the frequency that one class is predicted over the other). The main areas for future work are to better understand why these cost-sensitive learners perform sub-optimally and how this behavior can be remedied. Other areas for future work include analyzing additional cost-sensitive learners, analyzing data sets which exhibit more extreme class imbalance and analyzing additional data sets.

8. REFERENCES

[1] Domingos, P. Metacost: A General Method for Making Classifiers Cost-Sensitive. *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, 155-164, 1999.*

[2] Elkan, C. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 973-978, 2001.

[3] Hettich, S., Blake, C. and Merz, C. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Department of Information and Computer Science, 1998.

[4] Provost, F. Fawcett, T., and Kohavi, R. The case against accuracy estimation for comparing classifiers. *In Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[5] Quinlan, J. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, CA, 1993.

[6] Weiss, G. and Provost, F. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19: 315-354.

[7] Ciraco, M., Rogalewski, M., and Weiss, G. Appendix I: Additional results from improving classifier utility by altering the misclassification cost ratio, 2005.
http://storm.cis.fordham.edu/~gweiss/ubdm05-appendix.html