MetaCost: A General Method for Making Classifiers Cost-Sensitive

Pedro Domingos
Artificial Intelligence Group
Instituto Superior Técnico
Lisbon 1049-001, Portugal
pedrod@gia.ist.utl.pt
http://www.gia.ist.utl.pt/~pedrod

Abstract

Research in machine learning, statistics and related fields has produced a wide variety of algorithms for classification. However, most of these algorithms assume that all errors have the same cost, which is seldom the case in KDD problems. Individually making each classification learner cost-sensitive is laborious, and often non-trivial. In this paper we propose a principled method for making an arbitrary classifier cost-sensitive by wrapping a cost-minimizing procedure around it. This procedure, called MetaCost, treats the underlying classifier as a black box, requiring no knowledge of its functioning or change to it. Unlike stratification, MetaCost is applicable to any number of classes and to arbitrary cost matrices. Empirical trials on a large suite of benchmark databases show that MetaCost almost always produces large cost reductions compared to the cost-blind classifier used (C4.5RULES) and to two forms of stratification. Further tests identify the key components of MetaCost and those that can be varied without substantial loss. Experiments on a larger database indicate that MetaCost scales well.

1 Introduction

Classification is one of the primary tasks of data mining [18]. It has also been a subject of research in machine learning, statistics, pattern recognition, neural networks and other areas for several decades. As a result, many well-developed approaches to it now exist, including rule induction [20, 12], decision tree induction [8, 23], instance-based learning [11, 1], linear and neural classifiers [3], Bayesian learning [17, 16], and others. In classification problems, the goal is to correctly assign examples (typically described as vectors of attributes) to one of a finite number of classes. Most of the currently-available algorithms for classification are designed to minimize *zero-one loss* or *error rate*: the number of incorrect predictions made or, equivalently, the probability of making an incorrect prediction. This implicitly assumes that all errors are equally costly, but in most KDD applications this is far from the case. For example, 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. In general, misclassification costs may be described by an arbitrary cost matrix C, with $C(i,j)$ being the cost of predicting that an example belongs to class i when in fact it belongs to class j. The realization that in real-world applications non-uniform costs are the rule rather than the exception has led in recent years to an increased interest in algorithms for cost-sensitive classification. (Some of these will be discussed in the section on related work; Turney [25] provides an online bibliography on the topic.) Substantial work has gone into making individual algorithms cost-sensitive. Doing this for all algorithms available in the literature would be a very time-consuming enterprise, and often it is far from obvious how best to perform the conversion. A potentially better solution would be to have a procedure that converted a broad variety of error-based classifiers into cost-sensitive ones. To our knowledge, the only currently available procedure of this type is stratification—changing the frequency of classes in the training data in proportion to their cost [8, 9, 22]. However, this approach has several shortcomings. It distorts the distribution of examples, which may seriously affect the performance of some algorithms. It reduces the data available for learning, if stratification is carried out by undersampling. It increases learning time, if it is done by oversampling. Most seriously, it is only applicable to two-class problems and to multiclass problems with a particular type of cost matrix, one where $C(i,j) = C(j)$ (i.e., where the cost of misclassifying an example is independent of the predicted class).

In this paper we propose a new procedure for cost-sensitive classification that attenuates or eliminates the disadvantages of stratification. This procedure, called MetaCost, is based on wrapping a "meta-learning" stage around the error-based classifier in such a way that the classifier effectively minimizes cost while seeking to minimize zero-one loss. The next section describes in detail MetaCost and the reasoning leading to it. The following section describes an extensive empirical evaluation of MetaCost, where it is compared with stratification and where its properties, in particular its scalability, are investigated. The paper concludes with a discussion of related research, directions for future work, and a summary of results.

2 The MetaCost Algorithm

In order to obtain a procedure for making error-based classifiers cost-sensitive, let us start with some basic notions. If, for a given example x, we know the probability of each class j $P(j|x)$, the *Bayes optimal* prediction for x is the class i that minimizes the *conditional risk* [17]:

$$R(i|x) = \sum_j P(j|x)C(i,j) \qquad (1)$$

The conditional risk $R(i|x)$ is the expected cost of predicting that x belongs to class i. The Bayes optimal prediction is guaranteed to achieve the lowest possible overall cost (i.e., the lowest expected cost over all possible examples x, weighted by their probabilities $P(x)$). $C(i,j)$ and $P(j|x)$ together with the rule above imply a partition of the example space X into j (possibly nonconvex) regions, such that class j is the optimal (least-cost) prediction in region j. The goal of cost-sensitive classification is to find the frontiers between these regions, explicitly or implicitly. This is complicated by their dependence on the cost matrix C: in general, as misclassifying examples of class j becomes more expensive relative to misclassifying others, the region where j should be predicted will expand at the expense of the regions of other classes, even if the class probabilities $P(j|x)$ remain unchanged. In fact, we do not know what the optimal predictions are *even for the pre-classified examples in the training set*; depending on the cost matrix, they may or may not coincide with the classes that the examples are labeled with. If the examples in the training set were relabeled with their optimal classes according to the cost matrix given, an error-based classifier could be applied to learn the optimal frontiers, because the examples would now be labeled according to those frontiers. In the large-sample limit, a consistent error-based learner would learn the optimal, cost-minimizing frontiers. With a finite sample, the learner should in principle do no worse at finding these frontiers than it would at finding the optimal zero-one loss frontiers given the original training set.

The MetaCost procedure is based on this idea. In order to relabel the training examples with their "optimal" classes, we need to find a way to estimate their class probabilities $P(j|x)$. Note that this is different from finding class probabilities for unseen examples, and that the quality of these estimates is important only insofar as it influences the final frontiers produced; probability estimates can be quite poor and still lead to optimal classification, as long as the class that minimizes conditional risk given the estimated probabilities is the same that minimizes it given the true ones [16]. One possibility would be to use standard probability estimation techniques, such as kernel density estimation [17]. However, successful learning of a cost-sensitive classifier using this approach would require that the machine learning bias (i.e., the implicit assumptions) of both the classifier and the probability estimator be valid for the application domain. Strictly speaking, this is impossible unless the classifier and the density estimator are the same, and a mismatch between probability estimation and classification stages has indeed been found to hurt performance in a context similar to the present one ([13]; see also related work section below). For example, decision tree and rule inducers are some of the most effective learners for very-high-dimensional domains like those often found in KDD, but these are precisely those domains were commonly-used probability estimation techniques like kernel densities and mixture models are least effective. Our assumption here will be that the user has chosen a particular classifier because its characteristics are well suited to the domain, and that we should therefore also use that classifier and no other.

Many classifiers yield class probability estimates as a by-product of learning, but these are often very poor. For example, most decision tree and rule learners work by attempting to drive class probabilities to zero or one within each leaf or rule, and the resulting estimates are correspondingly off [7]. Because of this, and because some classifiers may not produce class probabilities, MetaCost allows their use, but does not require it. A more robust and generally-applicable method for obtaining class probability estimates from a classifier is suggested by recent research on model ensembles [10, 14]. Many authors (e.g, Breiman [5]) have found that most modern learners are highly unstable, in that applying them to slightly different training sets tends to produce very different models and correspondingly different predictions for the same examples, while the overall accuracy remains broadly unchanged. This accuracy can be much improved by learning several models in this way (or using other variations) and then combining their predictions, for example by voting. Thus MetaCost estimates class probabilities by learning multiple classifiers and, for each example, using each class's fraction of the total vote as an estimate of its probability given the example. (Not all learners are unstable in the fashion described; methods for applying

MetaCost to such learners are discussed in the section on future work.) Specifically, MetaCost uses a variant of Breiman's [5] *bagging* as the ensemble method. In the bagging procedure, given a training set of size s, a "bootstrap" resample of it is constructed by taking s samples *with replacement* from the training set. Thus a new training set of the same size is produced, where each of the original examples may appear once, more than once, or not at all. This procedure is repeated m times, and the resulting m models are aggregated by uniform voting (i.e., when an unclassified example is presented, the ensemble labels it with the class that is predicted by the greatest number of models). MetaCost differs from bagging in that the number n of examples in each resample may be smaller than the training set size s. This allows it to be more efficient. If the classifier being used produces class probabilities, a class's vote is estimated as the unweighted average of its probabilities given the models and the example. Also, when estimating class probabilities for a given training example x, MetaCost allows taking all the models generated into consideration, or only those that were learned on resamples the example was not included in. The first type of estimate is likely to have lower variance, because it is based on a larger number of samples, while the second is likely to have lower statistical bias, because it is not influenced by the example's own class in the training set.

In short, MetaCost works by: forming multiple bootstrap replicates of the training set, and learning a classifier on each; estimating each class's probability for each example by the fraction of votes that it receives from the ensemble; using Equation 1 to relabel each training example with the estimated optimal class; and reapplying the classifier to the relabeled training set. Pseudo-code for the MetaCost procedure is shown in Table 1. Note that, if the cost matrix changes, only the final learning stage needs to be repeated, and this is equivalent to a single run of the error-based classifier.

Table 1: The MetaCost algorithm.

Inputs:
- S is the training set,
- L is a classification learning algorithm,
- C is a cost matrix,
- m is the number of resamples to generate,
- n is the number of examples in each resample,
- p is *True* iff L produces class probabilities,
- q is *True* iff all resamples are to be used for each example.

Procedure MetaCost (S, L, C, m, n, p, q)

For $i = 1$ to m
  Let S_i be a resample of S with n examples.
  Let M_i = Model produced by applying L to S_i.

For each example x in S
  For each class j
    Let $P(j|x) = \frac{1}{\sum_i 1} \sum_i P(j|x, M_i)$
    Where
      If p then $P(j|x, M_i)$ is produced by M_i
      Else $P(j|x, M_i) = 1$ for the class predicted by M_i for x, and 0 for all others.
      If q then i ranges over all M_i
      Else i ranges over all M_i such that $x \notin S_i$.
  Let x's class = $\text{argmin}_i \sum_j P(j|x) C(i,j)$.

Let M = Model produced by applying L to S.

Return M.

3 Empirical Evaluation

The question of whether MetaCost reduces cost compared to the error-based classifier and to stratification was studied empirically using 28 benchmark databases from the UCI repository [4]. The C4.5 decision tree learner [23] was used as the error-based classifier because of its *de facto* role as a standard for empirical comparisons. The C4.5RULES post-processor, which converts C4.5's decision trees to sets of "IF ... THEN ..." rules, was also used, since it tends to improve accuracy and produces simpler, more comprehensible results [23]. In what follows, the C4.5–C4.5RULES combination will be referred to by the abbreviation "C4.5R." Except where noted, all experiments were carried out by randomly selecting 2/3 of the examples in the database for training the classifiers, and using the remaining 1/3 for measuring the cost of their predictions. The results reported are the average of 20 such runs. We first report the results of experiments on 15 multiclass databases, followed by experiments on 12 two-class databases. Lesion studies and a scaling-up study using a larger database complete this section.

3.1 Multiclass Problems

Experiments were conducted with two different types of cost model. In the first, each $C(i,i)$ was chosen at random from a uniform distribution in the $[0, 1000]$ interval, and each $C(i,j)$ for $i \neq j$ was chosen at random from the fixed interval $[0, 10000]$. Different costs were generated for each of the 20 runs conducted on each database; thus the standard deviations reported incorporate the effects of varying the cost matrix. In the second experiment, each $C(i,i)$ was chosen as before,

Table 2: Average costs and their standard deviations for multiclass problems.

Database	Costs from fixed interval				Costs from class-prob.-dependent interval			
	C4.5R	Underspl	Overspl	MetaCost	C4.5R	Underspl	Overspl	MetaCost
Annealing	1061±23	1076±18	989±20	984±24	1258±89	711±44	1027±53	46±3
Audiology	1842±90	2008±81	1569±83	1769±78	2264±195	609±124	1833±145	17±10
Glass	1986±81	1856±102	1786±90	1217±64	1169±92	548±36	1052±86	221±10
Iris	652±30	618±26	641±25	513±16	470±22	454±18	469±20	345±20
LED	2016±92	2181±78	1897±81	1484±67	835±44	814±52	727±21	393±16
Lenses	1624±132	1687±159	1596±167	1515±110	1171±264	910±218	972±115	192±18
Lung cancer	2363±261	1714±148	1957±263	1577±144	969±135	624±74	1013±146	370±39
Lymphogr.	1055±50	1000±52	970±57	721±33	1118±110	489±86	1054±111	60±5
Post-oper.	1239±152	686±31	1614±91	666±31	2322±230	841±112	2016±151	83±9
Pr. tumor	3475±88	3478±123	3759±98	3446±87	3193±338	936±117	1691±117	17±7
Solar flare	1602±88	1430±94	1425±79	875±43	1342±98	542±66	996±70	133±7
Soybean	716±35	718±44	739±46	873±91	682±33	632±37	685±35	283±26
Splice	659±12	602±13	715±10	556±11	412±12	424±9	419±11	342±4
Wine	685±24	647±33	670±29	558±25	461±18	475±13	455±14	264±13
Zoology	1148±103	1153±105	857±49	873±66	624±85	650±108	540±62	169±14

but $C(i,j)$ was chosen with uniform probability from the interval $[0,\ 2000\,P(i)/P(j)]$, where $P(i)$ and $P(j)$ are the probabilities of occurrence of classes i and j in the training set. Thus the expected value of $C(i,j)$ was $1000\,P(i)/P(j)$. This means that the highest costs are for misclassifying a rare class as a frequent one, and inversely for the lowest. This mimics the situation often found in practice (e.g., in the database marketing domains mentioned before) where the rarest classes are the ones that it is most important to identify correctly. When this is the case, a low error rate can be achieved simply by ignoring the minority classes, but the cost will be high. Cost-sensitive learning is thus particularly important in these problems. As before, a different cost matrix was generated for each run.

Since stratification cannot be directly applied to arbitrary multiclass cost matrices, we followed Breiman et al.'s [8] suggestion of making $C(j) = \sum_i C(i|j)$, where $C(j)$ is the cost of misclassifying an example of class j, irrespective of the class predicted. The training set was then resampled so as to make each class's probability equal to $P'(j) = C(j)P(j)/\sum_j C(j)P(j)$. (See [8, pp. 112-115] for a detailed justification of this procedure.) This was done in two different ways: by undersampling and by oversampling. In the undersampling procedure, all examples of the class j with highest $P'(j)$ are retained, and a fraction $P'(i)/P'(j)$ of the examples of each other class i is chosen at random for inclusion in the resampled training set. Although this is probably the most frequently used type of cost-based stratification, it has the disadvantage of reducing the data available for learning, which may increase cost. Thus oversampling is sometimes used instead. In this alternative, all examples of the class j with *lowest* $P'(j)$ are retained, and then the examples of every other class i are duplicated approximately $P'(i)/P'(j)$ times in the training set. This avoids the loss of training data, but may significantly increase learning time, particularly for superlinear algorithms.

MetaCost was applied considering for each example only the models it was not used to train, using C4.5R's rule class probabilities, and generating 50 resamples, each of size equal to the original training set's (i.e., $q =$ *False*, $p =$ *True*, $m = 50$ and $n = s$ in Table 1). The results obtained are shown in Table 2, and graphically in Figure 1. (Results obtained using other variants of MetaCost are reported in the section on lesion studies.) In the fixed-interval case, neither form of stratification is very effective in reducing costs, which is perhaps not surprising given that the approximations made in order to apply them are far from true. In contrast, MetaCost reduces costs compared to C4.5R and undersampling in all but one database, and compared to oversampling in all but three. In the probability-dependent case, which more closely matches the assumptions used to apply stratification, both undersampling and oversampling reduce cost compared to C4.5R in 12 of the 15 databases. MetaCost does better, achieving lower costs than C4.5R and both forms of stratification in all 15 databases. Globally, the average cost reduction obtained by MetaCost compared to C4.5R is approximately twice as large as that obtained by undersampling, and five times that of oversampling. In both sets of experiments, the costs obtained by MetaCost are lower than those of each of the other three algorithms with confidences exceeding 99% using sign and Wilcoxon tests. These results support the conclusion that MetaCost is the cost-

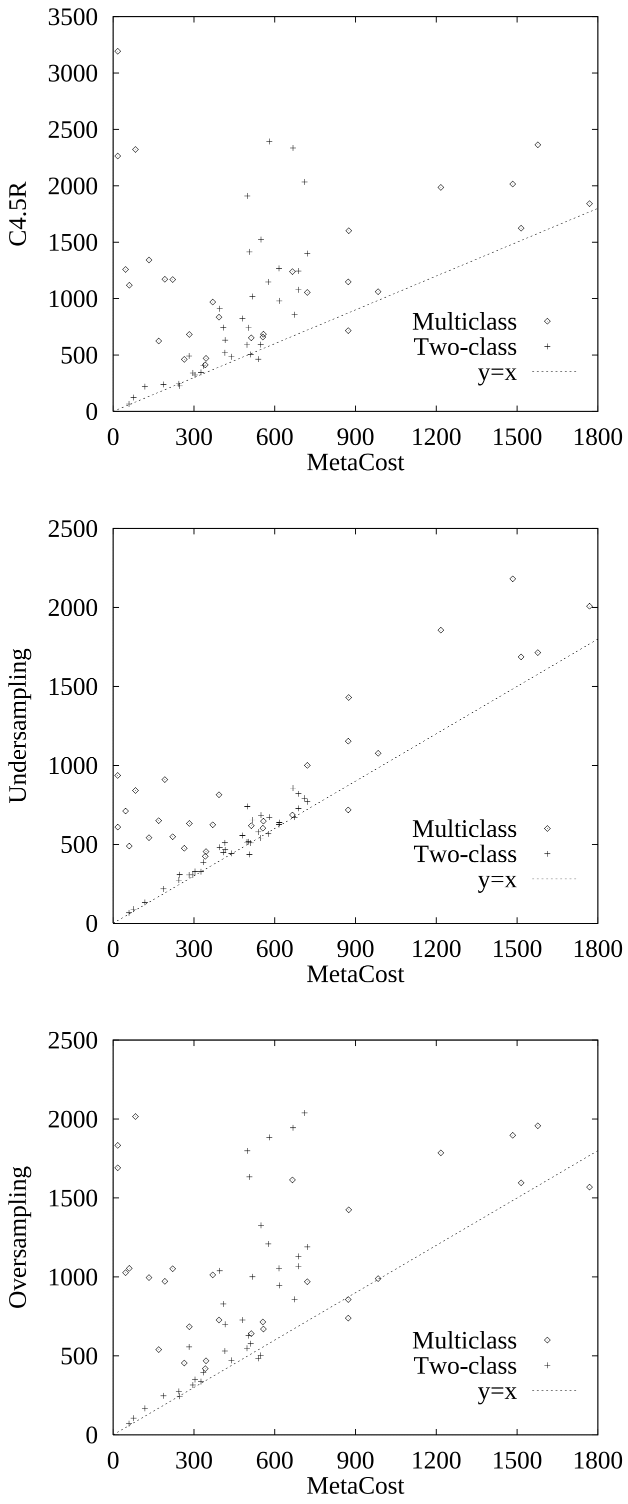

Figure 1: Comparison of MetaCost's costs (x axis) with those of C4.5R, undersampling and oversampling (y axes). Each point corresponds to a database and type of cost matrix. Points above the $y = x$ line are those where MetaCost outperformed the alternative classifier.

reduction method of choice for multiclass problems.

3.2 Two-Class Problems

In two-class problems where $C(1,1) = C(2,2) = 0$, stratification can be applied without any approximation by making $C(1) = C(2,1)$, $C(2) = C(1,2)$ and proceeding as before. Letting 1 be the minority class and 2 the majority class, experiments on two-class databases were conducted using the following cost model: $C(1,1) = C(2,2) = 0$; $C(1,2) = 1000$; $C(2,1) = 1000\,r$, where r was set alternately to 2, 5, and 10. Note that the absolute values of $C(2,1)$ and $C(1,2)$ are irrelevant for algorithm comparison purposes; only their ratio r is significant. The results obtained, using the same settings for MetaCost as before, are shown in Table 3, and graphically in Figure 1. Oversampling is not very effective in reducing cost with any of the cost ratios. Undersampling is effective for $r = 5$ and $r = 10$, but not for $r = 2$. MetaCost reduces costs compared to C4.5R, undersampling and oversampling on almost all databases, for all cost ratios. In all cases, the costs obtained by MetaCost are lower than those of each of the other three algorithms with confidences exceeding 99% using sign and Wilcoxon tests (except for the sign test for undersampling with $r = 10$, where the confidence is 98%). These results support the conclusion that MetaCost is the cost-reduction method of choice even for two-class problems where stratification can be applied without approximation.

3.3 Lesion Studies

Several questions arise in connection with MetaCost's results. How sensitive are they to the number of resamples used? Would it be enough to simply use the class probabilities produced by a single run of the error-based classifier on the full training set? Would MetaCost perform better if all models were used in relabeling an example, irrespective of whether the example was used to learn them or not? And how well would MetaCost do if the class probabilities produced by C4.5R were ignored, and the probability of a class was estimated simply as the fraction of models that predicted it? This section answers these questions by carrying out the relevant experiments. For the sake of space, only results on the two-class databases are presented; the results on multiclass databases were broadly similar. Table 4 reports the results obtained for $r = 2$, 5 and 10 by the following variations of MetaCost: using 20 and 10 resamples instead of 50 (labeled "m=20" and "m=10"); relabeling the training examples using the class probabilities produced by a single run of C4.5R on all the data (labeled "C4 Probs"); ignoring the class probabilities produced by C4.5R (labeled "0-1 Votes"); and using all models in relabeling an example (labeled "All Ms"). In each case,

Table 3: Average costs and their standard deviations for two-class problems.

Cost ratio	Database	C4.5R	Underspl	Overspl	MetaCost
2	Breast cancer	505±25	509±14	577±23	511±20
	Credit	239±11	218±8	247±10	187±7
	Diabetes	404±12	386±10	395±8	335±7
	Echocardiogram	590±23	514±21	549±22	497±22
	Heart disease	323±15	328±11	350±15	304±10
	Hepatitis	346±24	327±21	337±21	326±25
	Horse colic	244±10	273±14	275±13	244±8
	Labor	226±35	308±33	245±35	247±24
	Liver disease	593±24	540±15	503±18	548±16
	Promoters	340±29	306±30	316±35	296±16
	Sonar	484±25	442±21	472±33	439±21
	Voting	65±7	67±5	71±6	59±4
5	Breast cancer	1078±70	821±37	1130±57	688±21
	Credit	491±28	306±15	557±24	282±11
	Diabetes	824±34	556±20	727±21	480±13
	Echocardiogram	1244±65	727±43	1068±67	688±23
	Heart disease	632±43	466±23	700±37	416±22
	Hepatitis	741±60	517±47	628±54	503±40
	Horse colic	520±24	510±31	531±31	415±24
	Labor	463±89	579±48	484±54	539±35
	Liver disease	1268±75	625±22	1054±46	616±13
	Promoters	743±69	449±40	829±102	409±32
	Sonar	1019±66	654±49	1001±57	517±23
	Voting	123±18	89±9	106±13	76±6
10	Breast cancer	2034±148	792±37	2039±98	711±16
	Credit	911±57	481±29	1039±53	396±13
	Diabetes	1524±72	684±24	1326±63	549±15
	Echocardiogram	2335±137	856±60	1945±151	668±16
	Heart disease	1147±90	567±40	1209±83	576±17
	Hepatitis	1400±123	770±62	1190±110	721±51
	Horse colic	979±50	638±17	946±57	617±16
	Labor	858±179	671±22	858±95	674±34
	Liver disease	2393±162	671±33	1883±103	580±13
	Promoters	1414±138	436±12	1634±173	506±52
	Sonar	1910±138	740±69	1799±132	498±20
	Voting	220±37	132±16	168±27	118±8

all other settings are those used in the previous sections. Some of the main observations that can be made by comparing Table 4 with Table 3 are:

- In the $m = 50$ to $m = 10$ range, cost increases as the number of resamples decreases, but only very gradually. In particular, there is no significant difference between the costs obtained with $m = 50$ and $m = 20$, for all costs ratios, and with $m = 10$ MetaCost still reduces costs compared to C4.5R and both forms of stratification in almost all datasets, for all cost ratios.

- However, using multiple runs to estimate class probabilities is essential. Using a single run on all the data produces worse results than MetaCost and undersampling in almost all datasets for all cost ratios (except that it performs similarly to undersampling for $r = 2$). It still outperforms oversampling and C4.5R.

- Ignoring C4.5R's class probabilities increases cost in a majority of the datasets, but the relative differences are generally minor. MetaCost in this form still outperforms C4.5R and both types of stratification in a large majority of the datasets, for all cost ratios.

- Using all models decreases cost for $r = 10$ but increases it for $r = 5$ and $r = 2$. In all three

Table 4: Average costs and their standard deviations for different versions of MetaCost.

Cost ratio	Database	m=20	m=10	C4 Probs	0-1 Votes	All Ms
2	Breast cancer	514±18	506±25	498±22	495±19	519±19
	Credit	188±7	196±6	227±9	202±8	193±7
	Diabetes	343±7	348±8	390±9	357±8	355±8
	Echocardiogram	477±20	460±20	542±20	507±26	547±25
	Heart disease	292±12	306±10	320±14	308±12	300±11
	Hepatitis	330±24	333±25	337±22	317±22	336±21
	Horse colic	247±9	258±10	255±10	237±8	255±10
	Labor	213±29	300±35	242±33	266±29	208±26
	Liver disease	535±11	553±17	566±16	537±13	546±19
	Promoters	310±25	337±20	327±45	357±31	330±30
	Sonar	436±24	449±26	451±25	469±18	429±18
	Voting	58±5	65±4	64±6	59±4	60±5
5	Breast cancer	712±24	732±27	823±45	809±41	719±25
	Credit	277±12	289±12	331±20	287±14	286±11
	Diabetes	477±10	511±13	600±27	490±11	499±11
	Echocardiogram	699±31	684±32	876±55	791±47	730±32
	Heart disease	418±20	442±21	514±42	404±20	431±24
	Hepatitis	546±29	551±30	640±55	579±48	513±34
	Horse colic	405±23	417±17	530±26	446±28	453±20
	Labor	582±51	542±44	484±64	403±54	534±50
	Liver disease	613±14	646±20	751±33	707±29	628±16
	Promoters	473±40	456±46	569±90	491±41	357±25
	Sonar	556±31	638±38	834±59	608±37	552±41
	Voting	76±6	79±6	86±9	92±6	73±6
10	Breast cancer	733±17	707±25	765±49	981±44	700±19
	Credit	412±14	403±21	514±35	427±22	400±16
	Diabetes	556±15	577±16	688±33	623±21	551±10
	Echocardiogram	664±15	678±17	986±92	884±53	670±15
	Heart disease	560±26	576±22	596±43	514±39	529±15
	Hepatitis	667±38	710±60	942±97	709±64	675±49
	Horse colic	643±14	615±18	730±37	585±53	596±13
	Labor	666±29	718±46	682±72	645±82	634±31
	Liver disease	580±13	628±30	847±48	673±30	580±13
	Promoters	531±91	429±19	686±91	573±58	446±13
	Sonar	540±26	590±29	947±88	617±53	490±20
	Voting	105±11	106±10	106±13	118±9	106±10

cases the relative differences are generally minor, and the performance vs. C4.5R and stratification is generally similar.

To summarize, the one crucial element of MetaCost is the use of multiple runs to estimate class probabilities, but a number of runs as low as 10 is sufficient for excellent performance. The use of the error-based learner's class probabilities is beneficial but not critical, as is estimating an example's class probabilities excluding the models it was used to learn.

3.4 Scaling Up

An obvious potential disadvantage of MetaCost as used so far is that it increases learning time compared to the error-based classifier. In the databases used in the previous sections, where all learning times are on the order of seconds or fractions of a second, this is arguably immaterial. But in larger databases this might become a serious limitation. Although MetaCost only increases time by a fixed factor (the number of resamples, approximately) and so its asymptotic time complexity is of the same order as the error-based classifier's, the increased constant may be critical in large-scale applications. One solution lies in the fact that, since the multiple runs of the error-based classifier required to form the probability estimates are completely independent of each other, they can be trivially parallelized, reducing the time increase factor

Table 5: Costs and CPU times (in minutes and seconds) of C4.5R, oversampling and six variants of MetaCost on the shuttle database.

Algorithm	No Noise		10% Noise	
	Cost	Time	Cost	Time
C4.5R	15.6	1:41	2591.2	91:56
Undersampling	15.9	0:01	232.5	1:40
$m=10$, $n=\frac{s}{10}$	0.6	1:44	52.4	4:18
$m=10$, $n=\frac{s}{100}$	0.7	0:15	53.2	0:17
$m=20$, $n=\frac{s}{4}$	0.6	2:43	-	-
$m=20$, $n=\frac{s}{10}$	0.6	1:51	52.5	8:18
$m=50$, $n=s$	0.6	28:49	-	-
$m=50$, $n=\frac{s}{10}$	0.6	2:38	52.5	21:35

to about two. But a potential solution that does not require parallel processing is to use resamples that are smaller than the original training set (in addition to using a relatively small number of resamples, for example 10). Smaller resamples may result in higher costs, but conceivably still lower than those obtained with the error-based learner or with stratification, and therefore still worthwhile. Further, the increase in cost caused by learning the class probabilities on smaller samples may be offset or exceeded by the reduction obtained by the use of multiple models. Indeed, this idea is behind Breiman's [6] successful "pasting" method for scaling up learners. At the same time, reducing resample sizes will reduce running time, by a factor that will be particularly significant if the error-based learner used has superlinear running time. For example, if the classifier's running time is quadratic in the number of examples, using a tenth of the examples will reduce running time for each resample by a factor of 100. If 10 resamples are used, this will make the CPU time of the probability estimation phase an order of magnitude smaller than that of a single run of the error-based classifier on all the data, and therefore insignificant.

To test whether this approach is feasible, experiments were carried out using the largest database available in the UCI repository: shuttle [4]. The goal in this database is to diagnose the state of the space shuttle's radiators from a set of sensor readings. This problem is well suited for testing cost-sensitive algorithms, because there is a large majority of one class (the "normal" state) and it is easy to obtain very low error rates [12], but presumably the cost of missing one of the rare anomalous states is potentially much higher than that of a false alarm, making error rate an inappropriate measure of performance.

There are seven possible states, and nine numeric readings. The database contains 43500 examples from one shuttle flight and 14500 from another. The first flight was used for training, and the second for testing. All runs were carried out on a 300 MHz Pentium computer. Costs were set to $C(i,i) = 0$ for all i, and to $C(i,j) = 1000\,P(i)/P(j)$ for all $i \neq j$. It was not possible to obtain results for oversampling because C4.5R running on the expanded training set exceeded the available memory. (Judging from the results in the previous sections, there is a high probability it would do worse than undersampling.) Table 5 shows the costs and running times for C4.5R, oversampling, and MetaCost with several combinations of m (number of resamples) and n (resample size, as a function of the training set size s). As before, MetaCost was used with $p = \mathit{True}$ and $q = \mathit{False}$ (see Table 1). Undersampling is fast, but it does not reduce cost. MetaCost with 10 resamples each one tenth the size of the original training set reduces cost by over an order of magnitude, and its running time is very similar to C4.5R's. Increasing the number of resamples and the resample sizes predictably increases running time, but does not further reduce cost. Reducing resample size to 1/100 of the training set size increases cost by 17%.

The poor results obtained by undersampling suggest that MetaCost's excellent performance is not just due to the problem being "easy," requiring only a small number of examples to achieve low costs. However, as a further check we repeated the experiment with 10% class noise added to training and test sets (i.e., with 10% probability an example's class was changed to a different one, with all classes having the same probability of being the new one). The results are also shown in Table 5. Costs and running times are now much higher for all algorithms, and undersampling is now effective at reducing C4.5R's cost, but MetaCost is again by far the best performer. As before, increasing m above 10 produces no significant improvements. (For $n = \frac{s}{4}$ and $n = s$ C4.5R exceeded the available memory.) Remarkably, MetaCost with $m = 10$ and $n = \frac{s}{10}$ is over an order of magnitude faster than C4.5R, while reducing cost by over an order of magnitude. This result, which may appear surprising at first, is due to the fact that estimating class probabilities by averaging several models learned on small subsamples has the effect of filtering out noise, producing a cleaner dataset, on which C4.5R runs much faster than on the original one. Examining the algorithms' output shows that MetaCost induces a single short rule for each anomalous state and makes the normal state the default, while C4.5R's output is over an order of magnitude larger. C4.5R is known to be inefficient on large noisy databases [12], so these results may not generalize to other error-based learners. However, although preliminary, they are a strong indication that MetaCost can be applied effectively to large databases, reducing cost without significantly increasing running time. They also suggest that in noisy databases MetaCost may additionally be useful as a method for speeding up learning.

4 Related Work

Cost-sensitive learning is the subject of a burgeoning literature, which space does not allow us to review here. We point the reader to [15] for a brief review, and to [25] for an online bibliography. This section discusses those elements of previous research that are most closely related to MetaCost.

Chan and Stolfo [9] have proposed a variation of stratification that involves learning multiple classifiers on stratified subsamples of the database, which are then combined by a classifier that uses the others' outputs as inputs. This method does not produce a single model, and thus its output is hard to understand, while MetaCost produces a single model of similar complexity to that of the error-based classifier. Compared to stratification by undersampling, Chan and Stolfo's [9] method avoids the loss of training data, but is also only applicable (in its present form) to two-class problems. It is more *ad hoc* than stratification, lacking large-sample guarantees or clear foundations for its resampling scheme. Unlike MetaCost, it requires repeating all runs every time the cost matrix changes. It has only been tested in a single domain (credit-card fraud detection).

Ting and Zheng [24] have proposed a cost-sensitive variant of boosting (an ensemble method) for decision trees. It is significant here because it should be easily adaptable to other error-based learners, and like MetaCost creates a cost-sensitive learner from multiple runs of an error-based one. However, like Chan and Stolfo's [9] method, it does not produce a single comprehensible model. Compared to regular boosting, it is also more *ad hoc*, lacking its guarantees of near-optimal performance on the training data. It requires repeating all runs every time the cost matrix changes. Based on published results, it appears to produce substantially smaller cost reductions than MetaCost.

MetaCost's architecture is in some respects similar to that of CMM [13], a meta-learner that combines multiple models into a single one. While MetaCost's goal is to reduce costs, CMM's goal is to increase comprehensibility, while retaining some of the accuracy gains of multiple models. MetaCost uses the model ensemble to relabel training examples, while CMM uses it to label additional artificially-generated examples. A combination of the two may conceivably bring together the advantages of both.

5 Future Work

One item for future work is to carry out experiments on additional large databases, and using other error-based learners besides C4.5R. Of particular interest would be to apply MetaCost to algorithms that are not unstable with respect to variations in the training set, like k-nearest neighbor [11] and naive Bayes [16]. In its present form, MetaCost may not be very effective with these algorithms, but an alternative is readily suggested by the results of [2] and [26]. Their method consists of learning multiple models using different subsets of the attributes, instead of different subsets of the examples. K-nearest neighbor and naive Bayes are unstable with respect to changes in the attributes used, and this method was indeed found to reduce those algorithms' errors to an extent similar to bagging's with other learners. It is readily incorporated into MetaCost.

The application of MetaCost to large databases may be improved by stratifying the subsamples used, and/or by using partitioning instead of bagging. This would be similar to the first phase of Chan and Stolfo's [9] method, and might avoid losing useful information in some of the resamples. It might be necessary, however, to correct the probability estimates obtained for the effects of stratification.

An interesting comparison that has not yet been performed is to use an "off-the-shelf" probability estimator (e.g., kernel densities) for the first phase of MetaCost instead of multiple runs of the error-based classifier. Although unlikely to be generally useful, given previous results [13], this may be successful for some domains and some combinations of probability estimator and classifier. More generally, the effect on MetaCost's performance of the quality of the probability estimates used needs to be investigated. This is best done using synthetic data, for which we know the true class probabilities for every example. It would also be interesting to do an ROC analysis of MetaCost, by varying the probability thresholds at which an example's relabeling changes from one class to another [21].

The current version of MetaCost is based on bagging. An alternative method for constructing model ensembles is boosting [19]. Boosting often achieves lower error rates than bagging, and using it in MetaCost might produce corresponding reductions in cost.

6 Conclusion

KDD applications have often been hindered by the lack of powerful cost-sensitive learners. Converting individual error-based learners into cost-sensitive ones is a tedious and sometimes difficult process, but the general-purpose alternative of stratification is of limited applicability, and has a number of other disadvantages. In this paper we proposed and evaluated MetaCost, a new procedure for making error-based classifiers cost-sensitive. MetaCost is based on relabeling training examples with their estimated minimal-cost classes, and applying the error-based learner to the new training set. Experiments show that MetaCost systematically reduces cost compared to error-based classification and to stratification, often by large amounts, and that MetaCost can be efficiently applied to large databases.

Acknowledgements

This research was partly funded by the PRAXIS XXI program. The author is grateful to all those who provided the datasets used in the empirical study.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] S. D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. *Proc. 17th Intl. Conf. on Machine Learning*, pp. 37–45, Madison, WI, 1998.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[4] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases. Dept. of Information and Computer Science, University of California at Irvine, CA, 1999. http://www.ics.uci.edu/-~mlearn/MLRepository.html.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[6] L. Breiman. Pasting bites together for prediction in large data sets and on-line. Technical report, Statistics Dept., University of California at Berkeley, CA, 1996.

[7] L. Breiman. Out-of-bag estimation. Technical report, Statistics Dept., University of California at Berkeley, CA, 1998.

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

[9] 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.

[10] P. Chan, S. Stolfo, and D. Wolpert, editors. *Proc. AAAI-96 Wkshp. on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*. AAAI Press, Portland, OR, 1996.

[11] B. W. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[12] P. Domingos. Linear-time rule induction. *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 96–101, Portland, OR, 1996.

[13] P. Domingos. Knowledge acquisition from examples via multiple models. *Proc. 14th Intl. Conf. on Machine Learning*, pp. 98–106, Nashville, TN, 1997.

[14] P. Domingos. Why does bagging work? A Bayesian account and its implications. *Proc. 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 155–158, Newport Beach, CA, 1997.

[15] P. Domingos. How to get a free lunch: A simple cost model for machine learning applications. *Proc. AAAI-98/ICML-98 Wkshp. on the Methodology of Applying Machine Learning*, pp. 1–7, Madison, WI, 1998.

[16] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[17] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, NY, 1973.

[18] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pp. 1–34. AAAI Press, Menlo Park, CA, 1996.

[19] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. *Proc. 13th Intl. Conf. on Machine Learning*, pp. 148–156, Bari, Italy, 1996.

[20] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.

[21] F. Provost and T. Fawcett. Analysis and visualization of classifier performance. *Proc. 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 43–48, Newport Beach, CA, 1997.

[22] 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.

[23] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[24] K. M. Ting and Z. Zheng. Boosting trees for cost-sensitive classifications. *Proc. 10th European Conf. on Machine Learning*, pp. 191–195, Chemnitz, Germany, 1998.

[25] P. Turney. Cost-sensitive learning bibliography. Online bibliography, Institute for Information Technology of the National Research Council of Canada, Ottawa, Canada, 1997. http://-ai.iit.nrc.ca/bibliographies/cost-sensitive.html.

[26] Z. Zheng. Naive Bayesian classifier committees. *Proc. 10th European Conf. on Machine Learning*, pp. 196–207, Chemnitz, Germany, 1998.