Class Probability Estimation and Cost-Sensitive Classification Decisions

Dragos D. Margineantu

The Boeing Company, Adaptive Systems, Mathematics & Computing Technology
P.O.Box 3707, M/S 7L-66, Seattle, WA 98124-2207, USA
dragos.d.margineantu@boeing.com

Abstract. For a variety of applications, machine learning algorithms are required to construct models that minimize the total loss associated with the decisions, rather than the number of errors. One of the most efficient approaches to building models that are sensitive to non-uniform costs of errors is to first estimate the class probabilities of the unseen instances and then to make the decision based on both the computed probabilities and the loss function. Although all classification algorithms can be converted into algorithms for learning models that compute class probabilities, in many cases the computed estimates have proven to be inaccurate. As a result, there is a large research effort to improve the accuracy of the estimates computed by different algorithms. This paper presents a novel approach to cost-sensitive learning that addresses the problem of minimizing the actual cost of the decisions rather than improving the overall quality of the probability estimates. The decision making step for our methods is based on the distribution of the individual scores computed by classifiers that are built by different types of ensembles of decision trees. The new approach relies on statistics that measure the probability that the computed estimates are on one side or the other of the decision boundary, rather than trying to improve the quality of the estimates. The experimental analysis of the new algorithms that were developed based on our approach gives new insight into cost-sensitive decision making and shows that for some tasks, the new algorithms outperform some of the best probability-based algorithms for cost-sensitive learning.

1 Introduction

The general framework for supervised learning assumes that a set of labeled examples $\langle \mathbf{x}_i, y_i \rangle$ (called training data) is available, where $\mathbf{x}_i$ is a vector of continuous or discrete values called *attributes* and y_i is the *label* of $\mathbf{x}_i$. The framework further assumes that there exists an underlying, unknown function, $f(x) = y$ that maps the attribute vectors to the set of possible labels. A learned model outputs a hypothesis $h(x)$ which is an approximation of $f(x)$, and minimizes the expected loss on previously unseen examples. In the case of classification, the labels are elements of a discrete set of classes $\{1, 2, \ldots, K\}$.

The last few years have seen supervised learning and classification methods applied to an increasing variety of applications, such as fraud and intrusion detection, medical and biological data analysis, remotely-sensed image analysis, prediction of natural disasters, time series prediction, and text analysis.

While most of the research efforts in classification have studied algorithms that learn models that try to minimize the proportion of errors (mistakes) that are made (or, *0/1-loss*), for most (if not all) of the practical applications mentioned above, the learned classifiers are required to minimize a non-uniform loss function that is different from the *0/1-loss*. Indeed, in all practical situations different kinds of prediction errors have different costs and a more realistic performance measure of a classification system is the total loss, calculated as the sum of the costs of all errors made by the system.

The algorithms described in this paper assume that, for a K-class problem, a K-by-K loss matrix L is available at learning time. The contents of $L(i, j)$ specify the cost incurred when an example is predicted to be in class i when in fact it belongs to class j. We will further assume that L is stationary, that is, none of the values in L changes during the learning or the decision making process.

To illustrate the properties of a loss matrix, let us consider the 2×2 cost matrix shown in Table 1. In this example the loss matrix indicates that if an example is labeled by the classifier as to be in class 2 when in fact it belongs to class 1, there will be a loss of about 100.5, while labeling an example from class 2 as being in class 1 incurs only a cost of 3.0. Without loss of generality we can assume that the values in L represent dollar amounts (see [15] and [22] for more detailed discussions), and that there are no costs associated with correct decisions ([22] shows that a loss matrix can always be transformed into an equivalent one with zero values on the diagonal).

Conceptually, given the general supervised learning framework, there are three major types of strategies for cost-sensitive learning.

The most common practical approach to cost-sensitive classification is to manipulate the training data (i.e., modify its distribution) in order to make the 0/1-loss learning algorithm output a hypothesis that minimizes the costs of the decisions for future examples. For two-class problems, the simplest and most common way to do this is to present the learning algorithm with a training set in which the proportions of examples in the two classes are changed according

Table 1. Example of a loss matrix for a 2-class problem

	Correct Class	
Predicted Class	1	2
1	0.0	3.0
2	100.5	0.0

to the ratio of the cost values [8]. This procedure is called *stratification* (or *rebalancing*), and it is usually implemented by undersampling the examples from the less expensive class, or by oversampling the examples from the more expensive class [18]. Another method that uses training data manipulation to learn a cost-sensitive classifier is Domingos' MetaCost [11]. MetaCost is an algorithm that employs the learned class probability estimates for the training instances and relabels them. Then, it trains a 0/1-loss classification algorithm on the relabeled data to output the cost-sensitive hypothesis.

The second approach to the cost-sensitive learning problem is to change the internal mechanisms of the algorithm that compute the output hypothesis such that the algorithm will make use of the cost function (as an input parameter) to build the classifier [12,4,16,21,19].

Finally, the third approach uses the class probability estimates on the unseen (test) instances computed by the learned model. If the probabilities for each class given an example $\mathbf{x}$, $P(y_i|\mathbf{x})$ are available, $\mathbf{x}$ should be labeled with y_{opt}, the class that minimizes the conditional risk of the labeling decision [13,20]:

$$y_{opt} = \operatorname*{argmin}_{y \in Y} R(y|\mathbf{x}) = \operatorname*{argmin}_{y \in Y} \sum_{j=1}^{K} P(j|\mathbf{x}) L(y, j). \qquad (1)$$

For this strategy, there are two distinct steps: estimating the class probilities and making the decision. No information about the loss function is used in during the probability estimation process, and therefore, there is no need to retrain the models if the loss function changes.

It is important to observe that any loss matrix defines precise decision boundaries - points for which the minimum conditional risk is reached two or more classification deicisions. In general, for a K-class task the decision boundaries are defined by points x for which there exist at least two distinct class labels j and k such that (the class probabilities of x satisfy) $R(j|x) = R(k|x); R(i|x) \geq R(j|x), \forall i, 1 \leq i \leq K; \sum_{i=1}^{K} P(i|x) = 1$. For two-class problems (with classes 0 and 1) with loss matrices L having zero values on the diagonal, the decision boundary is defined by $\beta = P(0|x) = 1 - P(1|x) = \frac{L(0,1)}{L(0,1)+L(1,0)}$. If the estimate for an instance happens to be exactly on the decision boundary, the label of that instance is assigned by tossing a fair coin.

This paper presents a new approach to the cost-sensitive learning problem that relies on a learned probabilistic model, but with the specific target of minimizing the cost incurred by the decisions rather than attempting to improve the overall quality of the probabilities. To achieve this target, our methods compute confidence estimates for the class probabilities, and make the decisions based on those estimates.

The next section presents the challenges of using decision tree algorithms for learning probabilistic models. Section 3 describes the new methods for cost-sensitive learning and gives an overview of the random forest algorithms that are used. Section 4 presents an experimental analysis of the new methods. Section 5 summarizes the paper and draws the conclusions.

2 Decision Trees for Probability Estimation

Decision tree algorithms [8,27] are among the most popular tools for building classification models. Any decision tree D can be transformed into a class probability estimator. The probability estimate of class j for an arbitrary instance x is

$$P(j|x) = \frac{N_j(D_x)}{N(D_x)}, \tag{2}$$

where D_x is the leaf of tree D that is reached by x, $N(D_x)$ is the total number of training examples that are assigned to D_x, and $N_j(D_x)$ is the number of training examples belonging to class j that reach leaf D_x.

As noted by several researchers [29,5,26,6,25], the class probability estimates of the decision trees are poor. There are three major factors that cause this deficiency. First, the greedy induction mechanism that splits the data into smaller and smaller sets leads to probability estimates that are computed based on very small samples, and this leads to inaccurate estimates. Second, most of the existing decision-tree induction algorithms focus on minimizing the number misclassifications (through the purity-based heuristics) and on minimizing the size of the model (through the pruning procedure). This causes the learned models to compute class probabilities that are too extreme (i.e., close to 0.0 and 1.0), as in the example above, and therefore incorrect. The third factor is the shape of the decision tree hypotheses (piecewise linear decision boundaries). This kind of decision space assigns uniform probability values to points that are in the same region and will not differentiate between points that are closer to the boundary of the region and points that are farther from the boundary.

Lately, several researchers have addressed the problem of improving the probability estimates computed by decision trees and other classification methods. One solution [4,30] is to apply a Laplace correction (or Dirichlet prior) as follows

$$P(j|x) = \frac{N_j(D_x) + \lambda_j}{N(D_x) + \sum_{i=1}^{K} \lambda_i} \tag{3}$$

The Laplace correction [17,9] will smooth probability estimates that are too extreme because of the small size of the sample that reaches the leaf. This smoothing permits reducing the effects of the second cause for inaccurate estimates (extreme probabilities), described at the beginning of this section.

To handle the other two sources of inaccuracy of tree-based probability estimates, one of the most effective techniques has proven to be the averaging of the probabilities computed by multiple models generated by Bagging [8]. Each of the models is trained using a bootstrap replicate [14] of the training data. Provost and Domingos [25] have developed one of the best tree-based class probability estimation algorithms by combining Laplace correction and Bagging. They called resulting method Bagged Probability Estimation Trees (or, B-PETs).

3 Learned Probabilistic Models and Decision Making

The truth is that, while research efforts for improving the overall quality of probability estimates computed by different learning algorithms are worthwhile, making the best (cost-sensitive) classification decision does not always require highly accurate probabilities. For example, consider a (two-class) problem (for which the class labels are 0 and 1) that has a loss of fifty associated with mispredicting a positive (1) example and a loss of one associated with the opposite error. For an arbitrary instance x, any estimated probability value $P(1|\mathbf{x})$ that falls in the interval $(\frac{1.0}{51.0}, 1.0]$ will lead to diagnosing $\mathbf{x}$ as belonging to class 1. Therefore, if a system estimates that the likelihood of $P(1|\mathbf{x})$ falling in the interval $[0.8, 1.0]$ is 99%, we should be more confident classifying x into class 1, than in a situation in which the system estimates that the likelihood of $P(1|\mathbf{x}) > \frac{1.0}{51.0}$ is about the same as the likelihood of $P(1|\mathbf{x}) < \frac{1.0}{51.0}$.

In other words, accurate probability estimates are sufficient but not necessary. In order to minimize the costs associated with different decisions, it is important however to know how much confidence we can have in the computed class probabilities, and, if possible, to use the confidence estimates to make better decisions especially in the case of points that lie close to the decision boundary, or in the case of points that have a wide confidence interval for the probability estimates.

Given that we are dealing with *estimates* of a variable (the class probability), these observations have led us to combining estimates of the shape of the distribution of the (probability) estimates together with the loss function, for a decision making procedure.

Based on these observations, we propose the following decision making procedure for two-class problems. Let $\mathbf{x}$ be an arbitrary instance, and L the loss matrix. Let β $(0 \leq \beta \leq 1)$be the decision boundary defined by L. First, compute an estimate of the probability $\mathcal{P}(0|\mathbf{x}, L)$ that the learner will output a class probability estimate $P(0|\mathbf{x})$ that is smaller than β. Next, use the computed estimate to decide on the class of x by using Equation 1.

Training a series of probability estimators provides a good means to empirically estimate the distribution of the class probabilities for an arbitrary instance. In particular we use Bagging to compute the estimates.

The pseudo code of the procedure is presented in Table 2. We have called this generic procedure Confidence-based Probability Estimation (or C-PE) because it makes the classification decision based on the "confidence" in the probability estimates (given by the probability distribution of the probabilities) and their values relative to the decision boundary.

One way of computing the probability from line [7] of the code is to approximate it by the proportion of models whose estimate is smaller than β. The second possibility is to compute the normal approximation of the distribution of the estimates $\mathcal{N}(p)$ and to assign

$$\mathcal{P} := \int_0^\beta \mathcal{N}(p)dp.$$

Table 2. Pseudo code for the proposed algorithm for making cost-sensitive decisions with CONFIDENCE-BASED PROBABILITY ESTIMATION (C-PE)

```
Input: a set $S$, of $m$ labeled examples:
       $S =< (x_i, y_i), i = 1, 2, \ldots, m >$,
       labels $y_i \in Y = \{1, 2\}$,
       $\lambda$ (a learning algorithm that computes class probability estimates),
       $L$ (a loss matrix),
       $x$ (an unlabeled example),

[1] for $t = 1$ to $T$ do
[2]     $S_t :=$ (Bootstrap) sample of $S$;
[3]     $\theta :=$ Train $\lambda(S_t)$; // the learned model
[4]     $P_t(0|x) := \frac{N_0(\theta_x)}{N(\theta_x)}$; $P_t(1|x) := 1 - P_t(0|x)$;
[5] endfor
[6] $\beta :=$ DecisionBoundary$(L)$;
[7] $\mathcal{P}(0|x, L) := Pr(P(j|x) < \beta)$; $\mathcal{P}(1|x, L) := 1 - \mathcal{P}(0|x, L)$;
[8] endfor

Output: $h_{CPE}(x) = \operatorname*{argmin}_{y \in Y} \sum_{j=0}^{1} \mathcal{P}(j|\mathbf{x}, L) L(y, j)$
// the optimal prediction w. respect to $L$ and $\mathcal{P}$
```

As base learning algorithm λ we have used decision trees. However, given that we were not only interested in having unbiased estimates of the mean but also a good estimates of the variance of the computed probabilities, we have explored adding different sources of randomness to the original tree learning algorithm: random split selection, and random attribute selection. Breiman has proposed a unified view of these techniques under the name of Random Forests [7] and has analyzed them in the context of 0/1-loss classification.

In the case of *random splits*, during the tree learning procedure, instead of selecting the best potential split, the algorithm will choose a split at random from among the N best potential splits. This procedure was introduced by Dietterich [10] and used classification problems. For the *random attribute selection* procedure, at each node, a subset of size F of the attributes is selected at random and the best potential split (the one that gives the highest gain ratio) on those attributes is chosen. Amit and Geman [1] have first explored this technique.

4 Experimental Analysis

We have implemented the methods described in the previous section by using Quinlan's C4.5 decision tree learning algorithm [27] as base learner. The first implementation of C-PE (denoted as *Bag*) grows each tree using the standard procedure. The second implementation (*RS*) selects randomly in each node a split from among the ten best splits. The third implementation selects at each node a random subset of the attributes of size F. Two versions of this method were tested: *RA-1* ($F = 1$) and *RA-logN* ($F = log(N)$, where M is the number of attributes). Pruning was never used in the algorithms that were tested.

We have also implemented Provost and Domingos' B-PET algorithm to compare the decisions made by the C-PE methods (relying on $\mathcal{P}(y|\mathbf{x}, L)$) with the decisions that rely on class probability estimates ($P(y|\mathbf{x})$).

We have tested all algorithms on ten data sets (see Table 3). Except for the Donations-bin data set, all were drawn from the UC Irvine Repository [3]. Donations-bin is the binary version of the KDD Cup 1998 data [2] for which the goal is to determine whether a person has made a donation after a direct mail campaign. The format of the data is similar to the one used in other studies: seven attributes, 95412 instances for training and 96367 instances for testing.

Unfortunately, these data sets do not have associated loss matrices L. Therefore, we generated loss matrices at random according to some loss matrix models. Table 4 describes four loss models, M1 through M4. The second column of the table describes how the misclassification costs were generated for the off-diagonal elements of L. In all cases, the costs are drawn from a uniform distribution over some interval. The diagonal values are always 0.

Given that the new methods presented here were specifically designed to minimize the loss associated with the classification decisions, we have used the BDeltaCost paired test presented in [23]. Appendix A gives a more detailed description of the test.

We have chosen to use the BDeltaCost test rather than ROC methods because the ROC methods give an overall measure of the quality of the rankings, whereas in our case we needed a statistical test for comparing models when the loss matrix is known. In other words, we focus on the analysis of the quality of the decisions of the different models.

Table 3. Data sets studied in this paper

Name	Data Set Size	Evaluation Method
Donations-bin	95412/96367	test set
Breast cancer (Wis.)	699	10-fold xval
Breast cancer (Yug.)	286	10-fold xval
Hepatitis	155	10-fold xval
Horse colic	200	10-fold xval
King-rook vs. king-pawn	3196	10-fold xval
Labor negotiations	57	10-fold xval
Liver disease	345	10-fold xval
Sonar	208	10-fold xval
Voting records	435	10-fold xval

Table 4. The models employed for generating the loss matrices used in the experiments. Unif$[a, b]$ indicates a uniform distribution over the $[a, b]$ interval. The diagonal elements of the loss matrices are always zero

Loss Model	$L(i, j)$ $i \neq j$
M1	Unif$[0, 5]$
M2	Unif$[0, 7]$
M3	Unif$[0, 10]$
M4	Unif$[0, 20]$

Performance was evaluated either by 10-fold cross validation or by using a test set (as noted in Table 3). For each cost model we generated ten loss matrices, and performed 10-fold cross validation on the Irvine ML data sets. This gives us 10 (matrices) $\times$ 4 (models) $\times$ 10 (folds) = 400 runs of the algorithms for each Irvine ML data set. In the case of the Donations-bin data, the evaluation was performed on the test set, resulting in 80 runs. For each of the runs, we performed the BDeltaCost statistical test to determine whether the learned models had statistically significant different expected losses, based on the 95% confidence interval.

Initially we have set the number of bagging rounds to be $T = 100$.

We tested separately two versions of the C-PE algorithms. The first version (C-PE-counts) estimates $\mathcal{P}$ by using the counts of the individual computed class probabilities P on each side of the decision boundary. The results for the Donations-bin data are shown in Table 5. The results for the Irvine sets are presented in Table 6. Each cell of the tables represents the percentage of wins, ties, and losses (respectively) for the algorithms that are tested. For example, the cell in row RA-1, column B-PET from Table 6 indicates that when RA-1 and B-PET were compared, for 20.2% of the runs RA-1 outperformed B-PET, in 22.3% of the runs B-PET outperformed RA-1 and for 57.5% of the runs BDeltaCost could not reject the null hypothesis based on a 95% confidence interval.

The second version of our algorithms (C-PE-normal) estimates $\mathcal{P}$ by computing the normal approximation $\mathcal{N}$ of P. The results for the Donations-bin data are presented in Table 7. The results for the Irvine sets are shown in Table 8.

Next, we tested the influence of the size of the ensemble on the performance of the algorithms. We reran all experiments for $T = 50$, and $T = 200$. While, the quality of all C-PE decisions was slightly worse (compared to the B-PETs) for $T = 50$, it has improved for $T = 200$ only for the smaller Irvine data sets.

Table 5. Results on Donations-bin for C-PE-counts ($T = 100$)

	B-PET	Bag	RS	RA-1
RA-logN	20-75-5	20-75-5	20-80-0	20-80-0
RA-1	15-60-25	20-60-20	0-100-0	
RS	15-60-25	20-65-15		
Bag	0-80-20			

Table 6. Results on the UCI data sets for C-PE-counts ($T = 100$)

	B-PET	Bag	RS	RA-1
RA-logN	44.8-42.2-11	17.4-52.5-30.1	17-59.8-23.2	48.2-43.4-8.4
RA-1	20.2-57.5-22.3	9.3-48.9-41.8	6.3-45.1-48.6	
RS	42.1-48.7-9.2	22.6-51.9-25.5		
Bag	43.2-48.6-8.3			

Table 7. Results on Donations-bin for C-PE-normal ($T = 100$)

	B-PET	Bag	RS	RA-1
RA-logN	20-75-5	25-75-0	25-75-0	20-80-0
RA-1	15-60-25	20-60-20	5-95-0	
RS	15-60-25	20-60-20		
Bag	0-75-25			

Table 8. Results on the UCI data sets for C-PE-normal ($T = 100$)

	B-PET	Bag	RS	RA-1
RA-logN	45.9-43.1-11	20.7-48.6-30.7	25-56.8-18.2	44-47-9
RA-1	18.9-58.1-23	9.7-51-39.3	8.7-45.8-45.5	
RS	44.4-44.5-11.1	19.8-50.6-29.6		
Bag	46.3-46.2-7.5			

5 Summary and Conclusions

We have presented a new approach to cost-sensitive classification. The methods that we proposed make a decision not only based on an estimate of the mean of the probabilities computed by the models in the ensemble, but they employ the distribution of individual probability estimates of the classifiers together with the loss matrix. Instead of outputting the average of the individual estimates of the component classifiers the way B-PETs do, the C-PE algorithms compute an estimate of the distribution of class probabilities and makes a decision based on that estimate and the loss function. C-PE provides a mechanism to make accurate cost-sensitive decisions even if accurate class probability estimates are hard or impossible to compute (because of inherent deficiencies of the algorithms, or because of the distribution of the data). C-PE is sensitive not only to the loss function, but also to the hypothesis learned by the base algorithm.

In the case of the UCI data sets we can observe that, the *RA-logN*, *RS* and *Bag* versions of C-PE outperform the Bagged Probability Estimation Trees (B-PET). However in the case of the very large Donations data set, B-PET is marginally outperformed only by *RA-logN* and performs much better than *Bag*. This shows that for larger data sets, B-PET is able to compute more accurate probabability estimates $P(y|\mathbf{x})$, whereas in the case of smaller data sets the confidence-based estimates are better for different amounts of randomness.

If we were to rank the C-PE methods based on the amount of randomness that they add to the procedure, *RA-1* adds the largest amount, and the results show that this might lead to larger losses associated with the decisions. The best overall performance belongs to the *RA-logN* implementation of C-PE. This might be the case because it adds the right amount of randomness to the bagging procedure. It would be interesting to analyze the performance of *RS* for different values of the number of splits (among which the random selection is made).

The experiments also show that a larger value for T (the number of bagging rounds) helps improving the quality of the decisions on the smaller data sets.

6 Discussion

To our knowledge, the only decision making approach that has used a confidence measure for probability estimates was presented in the work of Pednault et al. [24].

Saar-Tsechansky and Provost [28] compute an estimate of the variance of the class probabilities for unlabeled examples to decide on the set of instances to be labeled next, within an active learning procedure.

Preliminary experiments show that combining C-PE with uncertainty sampling in a cost-sensitive active learning procedure improves in terms of the number of examples that are needed to achieve similar performance, over a an active learning procedure that relies on probability estimates computed by B-PETs.

Acknowledgement

I would like to thank Roberto Altschul, Foster Provost, Claudia Perlich, Tom Dietterich, Rodney Tjoelker, and Pedro Domingos for their comments and discussions on confidence-based decision making and cost-sensitive learning. I would also like to thank Ed Pednault for earlier discussions we had on this topic.

References

1. Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997. 275
2. S. D. Bay. The UCI KDD archive. University of California, Irvine, Dept. of Information and Computer Sciences, 1999. [`http://kdd.ics.uci.edu/`]. 276
3. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. [`http://www.ics.uci.edu/~mlearn/MLRepository.html`]. 276
4. J. P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. E. Brodley. Pruning decision trees with misclassification costs. In C. Nedellec and C. Rouveirol, editors, *Lecture Notes in Artificial Intelligence. Machine Learning: ECML-98, Tenth European Conference on Machine Learning, Proceedings*, volume 1398, pages 131–136, Berlin, New York, 1998. Springer Verlag. 272, 273
5. A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997. 273
6. L. Breiman. Out-of-bag estimation. Technical report, Department of Statistics, University of California, Berkeley, 1998. 273
7. L. Breiman. Random forests. Technical report, Department of Statistics, University of California, Berkeley, 2001. 275
8. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984. 272, 273

9. B. Cestnik. Estimating probabilities: A crucial task in machine learning. In L. C. Aiello, editor, *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149, London, 1990. Pitman Publishing. 273
10. T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):139–158, 2000. 275
11. P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164, New York, 1999. ACM Press. 272
12. C. Drummond and R. C. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Machine Learning: Proceedings of the Seventeenth International Conference*, pages 239–246, San Francisco, CA, 2000. Morgan Kaufmann. 272
13. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, Inc. - Interscience, second edition, 2000. 272
14. B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, 1993. 273, 281
15. C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 2001. 271
16. W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: Misclassification cost-sensitive boosting. In *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 97–105, San Francisco, 1999. Morgan Kaufmann. 272
17. I. J. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. M. I. T. Press, Cambridge, Mass., 1965. 273
18. N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, 2000. 272
19. M. Kukar and I. Kononenko. Cost-sensitive learning with neural networks. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, Chichester, NY, 1998. Wiley. 272
20. T. Leonard and J. S. J. Hsu. *Bayesian Methods, An Analysis for Statisticians and Interdisciplinary Researchers*. Cambridge University Press, 1999. 272
21. D. D. Margineantu. Building ensembles of classifiers for loss minimization. In M. Pourahmadi, editor, *Models, Predictions and Computing: Proceedings of the 31st Symposium on the Interface*, volume 31, pages 190–194. The Interface Foundation of North America, 1999. 272
22. D. D. Margineantu. Methods for cost-sensitive learning. Technical report, Department of Computer Science, Oregon State University, Corvallis, OR, 2001. 271
23. D. D. Margineantu and T. G. Dietterich. Bootstrap methods for the cost-sensitive evaluation of classifiers. In *Machine Learning: Proceedings of the Seventeenth International Conference*, pages 583–590, San Francisco, CA, 2000. Morgan Kaufmann. 276, 281
24. E. P. D. Pednault, B. K. Rosen, and C. Apte. The importance of estimation errors in cost-sensitive learning. In *Cost-Sensitive Learning Workshop Notes*, 2000. 279
25. F. Provost and P. Domingos. Well-trained PETs: Improving probability estimation trees. Technical Report IS-00-04, Stern School of Business, New York University, 2000. 273
26. F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing classifiers. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann, San Francisco, 1998. 273

27. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993. 273, 275
28. M. Saar-Tsechansky and F. Provost. Active learning for class probability estimation and ranking. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 911–917. AAAI Press/MIT Press, 2001. 279
29. P. Smyth, A. Gray, and U. Fayyad. Retrofitting decision tree classifiers using kernel density estimation. In *Machine Learning: Proceedings of the Twelvth International Conference*, pages 506–514, 1995. 273
30. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616, San Francisco, CA, 2001. Morgan Kaufmann. 273

A BDeltaCost

BDeltaCost is a paired test that computes a confidence interval for the expected difference in cost of two classifiers. The test is based on the idea of *bootstrap* [14], a computational method that is used for estimating the standard error of a parameter of an unknown distribution, based on a random sample S drawn from that distribution. The bootstrap works by drawing with replacement T samples for S, each consisting of a number of data values equal to the number of elements in S. The value parameter of interest is computed for each of these samples. The standard error is estimated by the sample standard deviations of the T replicates (also called *bootstrap replicates*).

In a similar way, BDeltaCost tests the null hypothesis H_0 that two classifiers have the same expected loss (on new test data) against the alternative hypothesis H_1 that the two classifiers have different losses. The test draws repeated samples of the data and calculates the differences in loss for the two classifiers, sorts the resulting values in ascending order and rejects the null hypothesis if 0 is not contained by the interval defined by the middle $c\%$ values, for a $c\%$ confidence interval (e.g. for a 95% confidence interval and $T = 1000$ the test will check the interval between the 26th and 975th value). The way the test has been designed, Laplace corrections can be used to correct for zero values (that occured because of the small size of the test set) in the confusion matrices.

Margineantu and Dietterich [23] have shown that the BDeltaCost test works better and gives tighter confidence intervals, than the standard tests based on the normal distribution.