1

Improved Class Probability Estimates from Decision Tree Models

Dragos D. Margineantu
Thomas G. Dietterich
Department of Computer Science, Oregon State University
{tgd,margindr}@cs.orst.edu

Abstract

Decision tree models typically give good classification decisions but poor probability estimates. In many applications, it is important to have good probability estimates as well. This paper introduces a new algorithm, Bagged Lazy Option Trees (B-LOTs), for constructing decision trees and compares it to an alternative, Bagged Probability Estimation Trees (B-PETs). The quality of the class probability estimates produced by the two methods is evaluated in two ways. First, we compare the ability of the two methods to make good classification decisions when the misclassification costs are asymmetric. Second, we compare the absolute accuracy of the estimates themselves. The experiments show that B-LOTs produce better decisions and more accurate probability estimates than B-PETs.

1.1 Introduction

The problem of statistical supervised learning is to construct a classifier f from a labeled training sample $\langle \mathbf{x}_i, y_i \rangle$, where each $\mathbf{x}_i$ is a vector of attributes (predictor variables), and $y_i \in \{1, \ldots, K\}$ is a class label. The classifier should accept new data points $\mathbf{x}$ and predict the corresponding class label $y = f(\mathbf{x})$. We assume that the training sample consists of independent and identically distributed draws from an unknown underlying distribution $\mathbf{P}(\mathbf{x}, y)$.

There are two fundamental approaches to statistical learning of classifiers. One approach is based on constructing a probabilistic model $\hat{\mathbf{P}}(\mathbf{x}, y)$ of the data and then applying statistical inference methods to determine the conditional class probability distribution $\hat{\mathbf{P}}(y|\mathbf{x})$. The well-known Naive

Bayes algorithm is an example of this approach, where the joint distribution $\hat{\mathbf{P}}(\mathbf{x}, y)$ is represented by a mixture model: $\hat{\mathbf{P}}(\mathbf{x}, y) = \sum_y \hat{\mathbf{P}}(y)\hat{\mathbf{P}}(\mathbf{x}|y)$ with one component for each class. Using such a mixture model, the class conditional distribution can be computed as

$$\hat{\mathbf{P}}(y|\mathbf{x}) = \frac{\hat{\mathbf{P}}(y)\hat{\mathbf{P}}(\mathbf{x}|y)}{\sum_{y'} \hat{\mathbf{P}}(y')\hat{\mathbf{P}}(y'|\mathbf{x})}.$$

The second approach to constructing classifiers is based on a direct representation of the classifier as a decision boundary. Decision tree methods [4, 24], for example, represent the decision boundary by a tree of tests on individual attribute values. For example, suppose that each $\mathbf{x}$ is a vector of real-valued attributes $\mathbf{x} = (x_1, \ldots, x_n)$ and $y \in \{1, 2\}$. Then a decision tree takes the form of a nested **if-then-else** statement such as the following:

```
if $x_2 > 3$ then
    if $x_1 > 4$ then
        $y = 1$
    else
        $y = 2$
else if $x_3 > 2$ then
        $y = 2$
    else
        $y = 1$
```

Experience across many application problems suggests that the direct decision-boundary approach usually gives more accurate classifiers than the probability distribution approach. There seem to be two reasons for this.

First, the probability distribution approach requires that we make assumptions about the form of the probability distribution $\mathbf{P}(\mathbf{x}, y)$ that generated the data. These assumptions are almost always incorrect. Sometimes the assumptions are only slightly violated; in other cases, the assumptions introduce major errors.

Second, and perhaps more important, the probabilistic modeling approach does not adapt the complexity of the model to the amount and complexity of the data. In machine learning, there is always a tripartite tradeoff operating between the amount of data, the complexity of the models that are fit to the data, and the error rate of the resulting fitted model [17]. If the complexity of the models is held constant, then when there is insufficient data, the fitted model will exhibit high variance. If there is more than enough data (and the model assumptions are not completely correct), then the model will exhibit significant bias. Decision-boundary algorithms that dynamically adjust the complexity of the model to the amount and complexity of the data—such as decision trees—provide a better way of managing this three-way tradeoff.

However, decision boundary methods are designed only to make decisions, not to produce estimates of the class probabilities $\mathbf{P}(y|\mathbf{x})$. In many applications, it is essential to have these probability estimates. In particular, in cost-sensitive learning problems, such as medical diagnosis, fraud detection, or computer intrusion detection, the cost of making a false positive error may be substantially different from the cost of making a false negative error. Let $C(j,k)$ be the cost of predicting that $\mathbf{x}$ belongs to class j when in fact it belongs to class k. According to decision theory, we should choose the class to minimize the expected cost:

$$y = \underset{j}{\operatorname{argmin}} \sum_k \mathbf{P}(k|\mathbf{x}) C(j,k). \tag{1.1}$$

To carry out this computation, we need an estimate of the class probabilities $\mathbf{P}(k|\mathbf{x})$. How can we obtain such estimates from decision trees?

Existing decision tree algorithms estimate $\mathbf{P}(k|\mathbf{x})$ separately in each leaf of the decision tree by computing the proportion of training data points belonging to class k that reach that leaf. There are three difficulties with this. First, the decision tree was constructed using these same data points, so the estimates tend to be too extreme (i.e., close to 0 or 1).

Second, as one proceeds from the root of the decision tree down to the leaves, the training data is split into smaller and smaller subsets. The result is that each class probability estimate is based on a small number of data points, and hence, can be very inaccurate.

Third, each leaf of the decision tree corresponds to a region of the $\mathbf{x}$ space, and this procedure assigns the same probability estimate to all points in the region. To see why this is a problem, suppose we are estimating the probability of heart disease (the class, y) given blood pressure (bp). Imagine that there is a leaf of the tree that corresponds to $bp > 160$ and that out of the 100 examples that reach this leaf, 90 have heart disease. Then the class probability is estimated as $\mathbf{P}(\text{heart disease}|\mathbf{x}) = 0.9$. But suppose that some of these data points include a mix of patients, some of whom have blood pressure of 161 and others have blood pressure of 250. Surely we want to predict a lower probability of heart disease for the patient with 161 than for the patient with 250.

From this review, it is clear that there is a need to develop improved methods for estimating class probabilities from decision-boundary learning algorithms. This paper presents and compares two methods for obtaining class probability estimates from decision trees. The first method, bagged probability estimation trees (B-PETs), was developed by Provost and Domingos [23]. The second, bagged lazy option trees (B-LOTs), was developed by the authors by extending previous work of Buntine [7], Breiman [5], Friedman, Kohavi, and Yun [16], and Kohavi and Kunz [21]. Other tree-based methods that can be extended to produce probability estimates include boosting [14, 15], random forests [6, 20, 12], and Bayesian CART [9, 11].

We evaluate the quality of the probability estimates produced by these two methods in two ways. First, we compare the predicted and true probabilities on a synthetic data set. Second, we compare the average misclassification costs of the two methods when applied to cost-sensitive learning problems. We employ a new bootstrap-based statistical test to determine which method (B-PET or B-LOT) gives lower expected misclassification cost across 16 data sets taken from the Irvine repository [2]. The results show that B-LOTs give more accurate probability estimates and lower expected misclassification costs. However, B-PETs appear to provide a better estimate of the *relative* class probabilities. Hence, if a way can be found to calibrate these relative rankings, B-PETs might also be made to provide good probability estimates. In either case, B-PETs and B-LOTs give much better performance than a single decision tree.

The remainder of this paper is organized as follows. First, we review the bagged probability estimation trees introduced by Provost and Domingos. Next, we describe our new method, the bagged lazy option trees. Finally, we present our experimental evaluations of the probability estimates and discuss their significance.

1.2 Probability Estimation Trees

Provost and Domingos [23] have developed an interesting approach to obtaining class probabilities from decision trees. They start with Quinlan's popular C4.5 decision tree algorithm [24] and modify it in four ways.

First, they turn off the pruning phase of C4.5. Without pruning, C4.5 will continue growing the tree (and subdividing the training data) until each leaf is pure (i.e., contains examples from only one class) or contains only 2 examples.

Second, they turn off C4.5's collapsing mechanism. At each internal node, C4.5 invokes itself recursively to grow a decision subtree. When that recursive call returns, C4.5 compares the error rate of resulting subtree (on the training data) with the error rate that would be obtained by replacing that subtree with a single leaf node. Hence, the effect of these first two changes is to simplify C4.5 by removing all pruning procedures.

Third, they modify C4.5's probability estimation mechanism so that instead of simply estimating $\hat{\mathbf{P}}(k|\mathbf{x})$ by the proportion of examples in the leaf that belong to class k, it applies a Laplace correction as follows:

$$\hat{\mathbf{P}}(k|\mathbf{x}) = \frac{N_k + \lambda_k}{N + \sum_{k=1}^{K} \lambda_k} \tag{1.2}$$

where N is the total number of training examples that reach the leaf, N_k is the number of examples from class k reaching the leaf, K is the number of classes, and λ_k is the prior for class y_k. In their experiments, they set λ_k uniformly to 1.0 for all $k = 1, \ldots, K$.

The effect of the Laplace correction [18, 8, 3] is to shrink the probability estimates toward $1/K$. For example, a leaf containing two examples (both from class $k = 1$) will estimate $\hat{\mathbf{P}}(1|\mathbf{x}) = (2+1)/(2+2) = 0.75$ when $K = 2$ instead of 1.0.

Finally, they apply an ensemble method known as Bagging [5]. Bagging constructs L decision trees and then averages their class probability estimates. Each decision tree is constructed by taking the original training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and constructing a bootstrap replicate data set S_ℓ by drawing N data points uniformly with replacement from S [13]. This training set S_ℓ is then given to the modified C4.5 procedure to produce a decision tree T_ℓ and a probability estimator $\hat{\mathbf{P}}_\ell(y|\mathbf{x})$. The final probability estimates are computed as

$$\hat{\mathbf{P}}(y|\mathbf{x}) = \frac{1}{L} \sum_{\ell=1}^{L} \hat{\mathbf{P}}_\ell(y|\mathbf{x}).$$

These modifications to C4.5 address two of the three causes of poor probability estimates identified in the introduction. The Laplace correction addresses the problem that the estimates in each leaf are based on small amounts of data. The combination of no pruning and Bagging addresses the problem that the probability estimates are constant throughout the region of $\mathbf{x}$ space corresponding to a single leaf. The only problem unaddressed is that the probability estimates are still based on the training data, which was also employed to construct the decision trees.

We explored the possibility of using out-of-bag estimates to address this last problem. The idea is to compute the estimates $\hat{\mathbf{P}}_\ell(y|\mathbf{x})$ by using the data points that were *not* included in training set S_ℓ (i.e., the so-called "out-of-bag" data points). Unfortunately, this did not work well, perhaps because there are even fewer "out-of-bag" data points than there are "in-bag" data points.

1.3 Lazy Option Trees

The decision tree literature contains many methods for improving over the basic C4.5 system. We have combined two of these methods—lazy tree growing and options—to obtain further improvements over B-PETs.

Standard decision tree algorithms can be viewed as being "eager" algorithms—that is, once they have the training data, they grow the decision tree. "Lazy" algorithms, in contrast, wait until the unlabeled test data point $\mathbf{x}_u$ is available before growing the decision tree. In this, they are analogous to the nearest-neighbor classifier, which waits until $\mathbf{x}_u$ is available and then bases its classification decision on the data points that are nearest to $\mathbf{x}_u$ according to some distance measure [19, 1, 10, 25]

The advantage of lazy learning is that it can focus on the neighborhood around the test point $\mathbf{x}_u$. In the case of probability estimates, a lazy learning algorithm can base its estimate $\hat{\mathbf{P}}(y|\mathbf{x}_u)$ on the data points in the neighborhood of $\mathbf{x}_u$ and thereby side-step the problem that the probability estimates in decision trees are constant within each leaf. The disadvantage of lazy learning is that the amount of computer time required to classify each data point can be significantly larger than for eager algorithms. This makes lazy learning impractical for some applications.

Lazy learning in decision trees was explored by Friedman et al. [16]. In their approach, which we imitate, a lazy tree actually consists of only a single path from the root of the tree down to a leaf. This is the path that will be taken by the test data point $\mathbf{x}_u$. Since the tree is only constructed to classify this one point, there is no need to expand the other branches of the tree. In addition, the choice of which attribute and value to test at each node of the tree can be made focusing only on the one resulting branch. In standard decision tree algorithms, the choice of splitting attribute and value is based on the average improvement of all branches of the decision tree. In some cases, the accuracy of one branch may actually be decreased in order to obtain a large increase in the accuracy of the other branch. Lazy trees do not need to make this compromise.

One risk of lazy trees is that the tests included in the decision tree are satisfied only by the test point $\mathbf{x}_u$. To reduce this risk, we restrict the form of the permitted tests. Let x_m be the value of the mth attribute of the test data point $\mathbf{x}_u$. We do not allow the test to directly refer to x_m. The test may have the form $x_j > \theta$, $x_j < \theta$, or $s_j \neq \theta$ as long as $\theta \neq x_m$.

To estimate the class probabilities at the leaf of the lazy tree, we do not apply a Laplace correction. Instead, we simply compute the proportion of training examples belonging to class k.

The other technique that we have included in our approach is options (see Buntine [7] and Kohavi and Kunz [21]). An option tree is a decision tree in which a given internal node in the tree is split in multiple ways (using multiple attribute-value tests, or "options"). The final probability estimate is the average of the estimates produced by each of these options.

Our lazy option tree algorithm selects in each node the best b_t tests with the highest accuracy (where accuracy is measured by the reduction in the estimated entropy of the class variable $H(y)$ from the parent to the child). To avoid redundancy in the tree, we check for and remove a particular pattern of tests (called the "cousin rule"). Let U and V be two tests (e.g., $x_j > \theta$ for some j and θ). Suppose that node A has two options. One option tests U and leads to child node B; the other option tests V and leads to child C. Now suppose that B chooses V as an option. Then we do not permit node C to choose U as an option, because this would create two identical paths (where U and V were true) in the option tree.

One final detail requires mention. Before choosing the options at a node, our algorithm assigns weights to the examples in each class in order to balance the weighted class frequencies.

Table 1.1 sketches the lazy option tree learning algorithm, and Figure 1.1 shows an example of a tree induced by the lazy option tree algorithm.

Table 1.1. The lazy option tree learning algorithm.

```
LazyOptionTree($S$,$\mathbf{x}_u$,$MaxTests$,$MinG$,$MinLeaf$)
**Input**: Training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, Unlabeled test example $\mathbf{x}_u$,
        Maximum number of tests in a node $MaxTests$,
        Minimum gain for a test $MinG$,
        Minimum number of examples in a leaf $MinLeaf$,
    **for** $k = 1$ **to** $K$ **do**
        **assign** weights $\mathbf{w}$ such that $\sum_{i|y_i=k} w_i = 1$
    **choose** a set of $T$ tests of the form $x_j > \theta$, $x_j < \theta$, or $x_j \neq \theta$ **such that**
            $T < MaxTests$
            if best test has gain $g$, each test has a gain $> MinG \cdot g$,
            at least $MinLeaf$ examples satisfy the test, and
            the set of tests contains no tests ruled out by the cousin rule
    **if** no such tests exist **then** terminate with leaf node
    **else** call LazyOptionTree on elements of $S$ that satisfy the test
**end** LazyOptionTree
```

The options represent an alternative to the averaging mechanism of Bagging for improving the probability estimates of the decision trees. The advantage of the options mechanism is that tests having an information gain almost as high as the best test will be performed, while they might never be performed in decision trees, lazy trees, or even bagged trees. This increases the diversity of the ensemble and leads to better probability estimates.

Nonetheless, we found experimentally that the probability estimates can be further improved by employing bagging in addition to options. This completes our bagged lazy option trees (B-LOTs) algorithm.

1.4 Parameter Settings

To apply the B-PET method, the user must specify only two parameters: the minimum number of examples $MinLeaf$ permitted in each leaf of the tree, and the number of iterations L of bagging. To apply B-LOTs, the user must set $MinLeaf$, L, and also two more parameters: $MaxTests$—the maximum number of tests that are allowed in a node—and $MinG$—the gain proportion ($0.0 < MinG < 1.0$), a number indicating the minimum

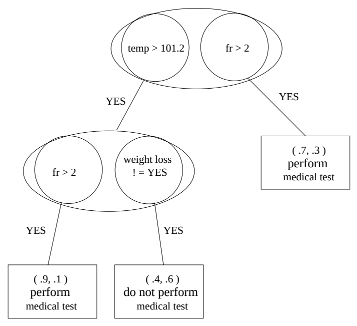

Figure 1.1. Example of a tree built by the lazy option tree algorithm for a specific unlabeled instance $\langle fr = 6$, $weight\ loss = NO$, $blood\ pressure = HIGH$, $body\ temperature = 101.5\rangle$. The task is to decide whether a patient is suspect of having AIDS, and therefore whether to perform an HIV test (fr stands for the frequency of recurrent infections per month, *temp* is the body temperature). The numbers in the leaves indicate the proportion of training examples that reach that leaf. Each leaf computes a probability estimate based on those numbers. These estimates are then averaged to produce the final estimates. The labels of the leaves indicate the decision in the case of 0/1 loss.

gain for a test in order to be selected. $MinG$ is the proportion out of the gain of the best test achieved within that node (e.g., if the best test in a node has a gain on g, we will discard all tests that have a gain of less than $MinG \cdot g$, within that same node).

1.5 Experiment 1: Probability Estimates on Synthetic Data

Figure 1.2 shows two one-dimensional gaussian distributions corresponding to $\mathbf{P}(\mathbf{x}|y = 1)$ and $\mathbf{P}(\mathbf{x}|y = 2)$. Let $\mathbf{P}(y = 1) = \mathbf{P}(y = 2) = 0.5$ be the probability of generating examples from classes 1 and 2, respectively. We constructed this synthetic problem so that we could measure the accuracy of the probability estimates produced by B-PETs and B-LOTs. We drew 1000 training examples from this joint distribution to form our training set S, which we then fed to the B-PET and B-LOT algorithms with the following parameter settings: $MinLeaf = 2$, $L = 30$, $MaxTests = 3$, and $MinG = 0.2$.

Figures 1.3 and 1.4 show scatter plots of the predicted and true probability values for the two methods. Perfect probability estimates would lie on the diagonal line shown in the plots. The scatter plot for the B-PETs shows that the points increase almost monotonically (with some errors), which indicates that the B-PETs have produced a good relative ranking of the test points. However, the absolute values of the probability estimates all

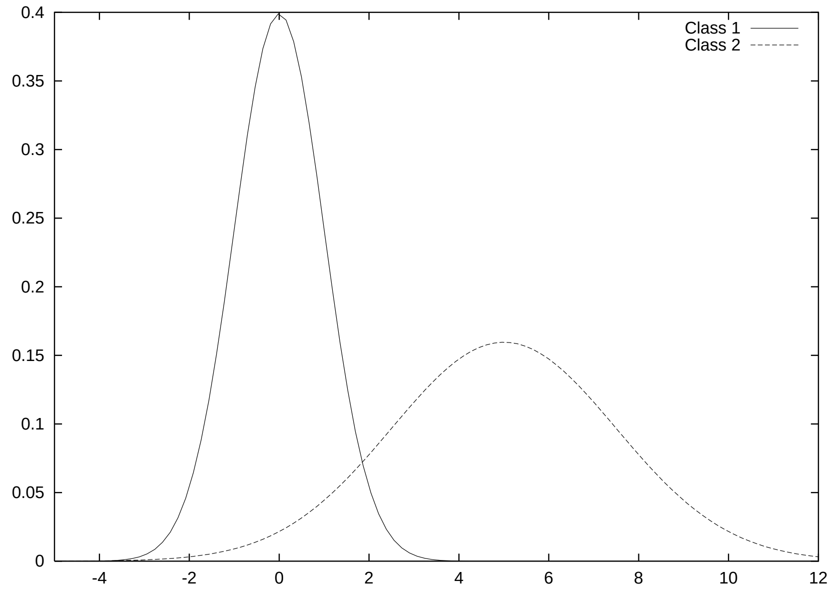

Figure 1.2. One-dimensional artificial data set

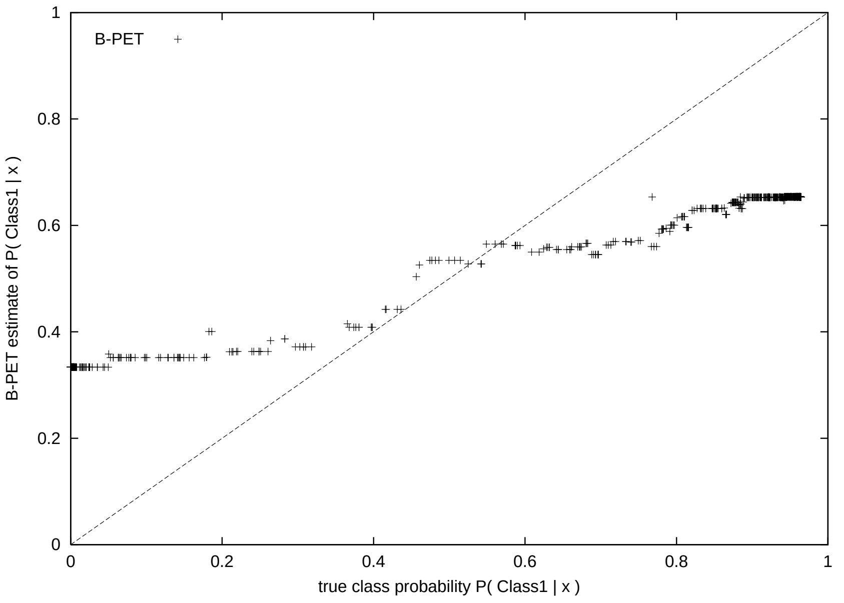

Figure 1.3. Scatterplot of B-PET probability estimates versus true probability values for 1000 test points

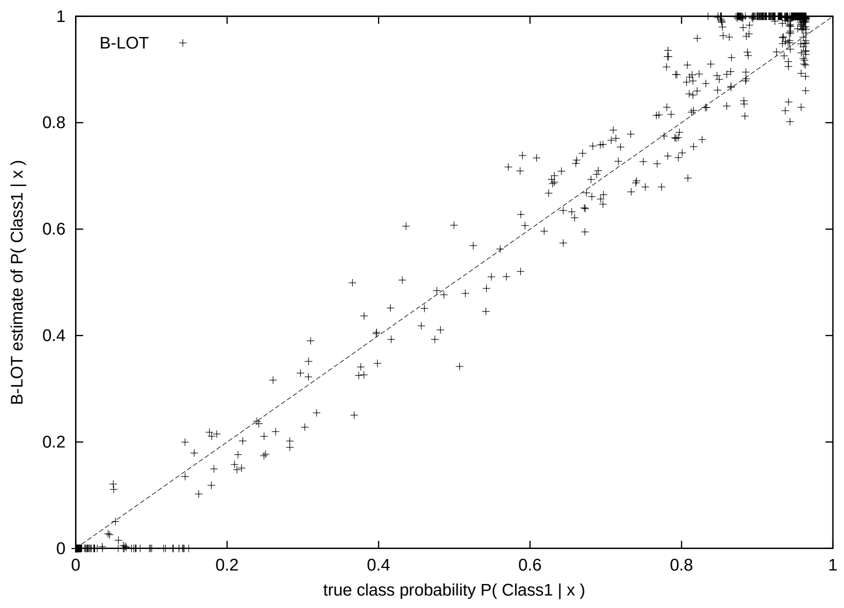

Figure 1.4. Scatterplot of B-LOT probability estimates versus true probability values for 1000 test points

lie in the range from 0.3 to 0.7, and the monotonic behavior is significantly non-linear. This behavior is partly a consequence of the Laplace shrinkage toward 0.5—however, if we remove the Laplace correction, the monotonic behavior of the curve is significantly degraded.

The plot for the B-LOTs, on the other hand, shows that the predicted probabilities cluster around the ideal diagonal and span the entire range from 0 to 1. However, individual points whose true probabilities are nearly identical can have predicted probabilities that differ by more than 0.2.

If this behavior holds in real domains, it would have two consequences. First, the B-PET estimates will not work well if they are plugged directly in to equation 1.1. This is because the given cost matrix C will determine an absolute probability threshold θ such that $\mathbf{x}_u$ should be classified as class 1 if $\hat{\mathbf{P}}(y|\mathbf{x}_u) > \theta$, and the absolute values of the B-PET predictions are poor. Consider, for example, if C dictates that $\theta = 0.2$. In such cases, all examples would be classified into class 1, with, presumably, high misclassification costs. In contrast, the B-LOT estimates will work quite well in equation 1.1, because they are well-calibrated. However, there will be several misclassified points, because of the variance in the estimates.

In 2-class problems, it is not necessary to use equation 1.1. Instead, cross-validation methods can be employed to determine experimentally a threshold that minimizes the expected misclassification cost on hold-out data points. In such cases, the B-PET method may work well unless the threshold happens to land exactly on one of the values for which the plot in Figure 1.3 is flat. Note that there are several such values, particularly at the more extreme probability values (which could be important if the misclassification costs are highly asymmetrical). For problems with more than 2 classes, cross-validation is difficult to employ, because separate thresholds must be determined for each pair of classes. Setting so many thresholds by cross-validation requires very large hold-out sets.

1.6 Experiment 2: Misclassification Errors on Benchmark Data Sets

A more realistic way of comparing B-PETs and B-LOTs is to evaluate their performance on a range of benchmark data sets drawn from real problems. We chose sixteen data sets from the UC Irvine ML repository [2] for this study. Unfortunately, most of the UC Irvine data sets do not have associated cost matrices C. Hence, we decided to generate cost matrices at random according to some cost matrix models.

Table 1.2 describes eight cost models, M1 through M8. The second column of the table describes how the misclassification costs were generated for the off-diagonal elements of C. In most cases, the costs are drawn from a uniform distribution over some interval. However, for models M5 and M6, the costs are distributed according to the class frequencies. In model M5, rare classes are given higher misclassification costs. This is characteristic of medical and fraud detection problems where a failure to detect a rare event (a serious disease or a fraudulent transaction) has a much higher cost than the failure to recognize a common event. In model M6, rare classes are given lower costs, to see if this makes any difference. The third column of Table 1.2 specifies how the diagonal elements of C are generated. Only models M7 and M8 have non-zero diagonal values.

For each cost model we generated twenty-five cost matrices and performed ten-fold cross validation. This gives us 25 (matrices) $\times$ 8 (models) $\times$ 10 (folds) = 2000 runs of the B-PET and B-LOT procedures for each data set. For each of these 2000 runs, we applied a statistical test, BDeltaCost, to determine whether the B-PETs and the B-LOTs had statistically significant different expected misclassification costs.

BDeltaCost is a bootstrap test developed by the authors [22] for comparing the cost of two classifiers γ_1 and γ_2. It works by first constructing a generalized confusion matrix M. The contents of cell $M(i_1, i_2, j)$ is the number of (test set) examples for which γ_1 predicted that they belong to class i_1, γ_2 predicted that they belong to class i_2, and their true class was j. BDeltaCost then constructs a three-dimensional cost matrix Δ such that $\Delta(i_1, i_2, j) = C(i_1, j) - C(i_2, j)$. In other words, the value of $\Delta(i_1, i_2, j)$ is the amount by which the cost of classifier γ_1 is greater than the cost of

Table 1.2. The cost models used for the experiments. Unif$[a, b]$ indicates a uniform distribution over the $[a, b]$ interval. $P(i)$ represents the prior probability of class i.

Cost Model	$C(i, j)$ $i \neq j$	$C(i, i)$
M1	Unif$[0, 10]$	0
M2	Unif$[0, 100]$	0
M3	Unif$[0, 1000]$	0
M4	Unif$[0, 10000]$	0
M5	Unif$[0, 1000 \times P(i)/P(j)]$	0
M6	Unif$[0, 1000 \times P(j)/P(i)]$	0
M7	Unif$[0, 10000]$	Unif$[0, 1000]$
M8	Unif$[0, 100]$	Unif$[0, 10]$

classifier γ_2 when γ_1 predicts class i_1, γ_2 predicts class i_2, and the true class is j. Given M and Δ, the difference in the costs of γ_1 and γ_2 can be computed by taking the "dot product":

$$M \cdot \Delta = \sum_{i_1, i_2, j} M(i_1, i_2, j)\Delta(i_1, i_2, j).$$

BDeltaCost computes a confidence interval for $M \cdot \Delta$ and rejects the null hypothesis that $M \cdot \Delta = 0$ if this confidence interval does not include zero. The confidence interval is constructed as follows. Normalize M by dividing by the number of test examples N, so that M's elements sum to 1. Treating M as a multinomial distribution, draw 1000 simulated confusion matrices $\tilde{M}_u$ by drawing N triples (i_1, i_2, j) according to this distribution. Compute the cost $\tilde{c}_u$ of each simulated confusion matrix (by computing $\tilde{M}_u \cdot \Delta$), sort these costs, and choose the 25th and 976th elements to construct the confidence interval.

The barchart in Figure 1.5 shows the results of the 2000 test runs for each data set. It plots the number of wins for B-LOTs, ties, and wins for B-PETs computed by BDeltaCost with a significance level of $\alpha = 0.05$ (i.e., 95% confidence). For all tasks except the lung-cancer domain, the number of wins for the B-LOTs is noticeably larger than the number of B-PET wins. In the case of nine domains (audiology, abalone, iris, lymphography, segmentation, soybean, voting-records, wine and zoo), the B-LOTs were the clear winner. These results suggest that the B-LOTs give better probability estimates in these application domains.

1.7 Concluding Remarks

Decision-boundary methods—as exemplified by decision tree learning algorithms—have proven to be very robust in many applications of machine learning. However, unlike probabilistic models, they do not provide estimates of $\mathbf{P}(y|\mathbf{x})$. Such probability estimates are essential for applica-

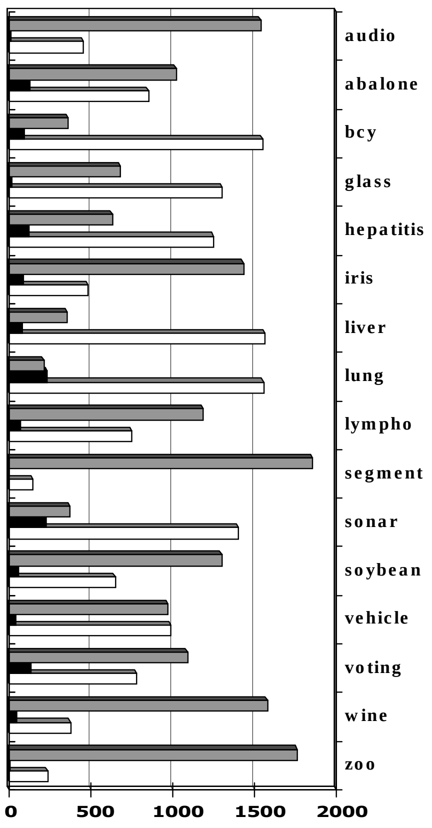

Figure 1.5. Barchart comparing the performance of B-LOTs and B-PETs for sixteen UCI data sets. White bars represent the number of ties, black bars represent the number of wins of B-PETs, and the grey bars represent the number of wins of B-LOTs computed by the BDeltaCost test.

tions that involve asymmetric loss functions. This paper has addressed the question of whether techniques can be found to compute good class probability estimates from decision trees.

The paper has introduced the bagged lazy option tree (B-LOT) method and compared it to the bagged probability estimate tree (B-PET) method. The results show experimentally that B-LOTs produce better-calibrated probability estimates and that these give lower expected misclassification cost across 16 benchmark datasets. However, the experiment with synthetic data showed that B-PETs do a very good job of predicting the relative probabilities of data points. So it may be possible to convert the B-PET estimates into more accurate probability estimates.

Based on the results from this paper, for applications where lazy learning is practical, bagged lazy option trees can be recommended as the best known approach to obtaining good class probability estimates and making good cost-sensitive classification decisions.

References

[1] B. K. Bhattacharya, G. T. Toussaint, and R. S. Poulsen. The application of Voronoi diagrams to non-parametric decision rules. In *Proceedings of the 16th Symposium on Computer Science and Statistics: The Interface*, pages 97–108, 1987.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] 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*, volume 1398, pages 131–136, Berlin, 1998. Springer Verlag.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Random forests. Technical report, Department of Statistics, University of California, Berkeley, CA, 1999.

[7] W. L. Buntine. *A theory of learning classification rules*. PhD thesis, University of Technology, School of Computing Science, Sydney, Australia, 1990.

[8] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In L. C. Aiello, editor, *Proceedings of the Ninthe European Conference on Artificial Intelligence*, pages 147–149. Pitman Publishing, 1990.

[9] H. Chipman, E. George, and R. McCulloch. Bayesian CART model search (with discussion). *Journal of the American Statistical Association*, 93:935–960, 1998.

[10] B. V. Dasarathy, editor. *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[11] D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. A Bayesian CART algorithm. *Biometrika*, 85:363–377, 1998.

[12] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):1, 2000.

[13] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap.* Chapman and Hall, New York, NY, 1993.

[14] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

[15] Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[16] Jerome H. Friedman, Ron Kohavi, and Yeogirl Yun. Lazy decision trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 717–724, San Francisco, CA, 1996. AAAI Press/MIT Press.

[17] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[18] I. J. Good. *The estimation of probabilities: An essay on modern Bayesian methods.* MIT Press, Cambridge, MA, 1965.

[19] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[20] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[21] Ron Kohavi and Clayton Kunz. Option decision trees with majority votes. In *Proc. 14th International Conference on Machine Learning*, pages 161–169. Morgan Kaufmann, 1997.

[22] D. D. Margineantu and T. G. Dietterich. Bootstrap methods for the cost-sensitive evaluation of classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, 2000. Morgan Kaufmann.

[23] Foster Provost and Pedro Domingos. Well-trained PETs: Improving probability estimation trees. Technical Report IS-00-04, Stern School of Business, New York University, 2000.

[24] J. R. Quinlan. *C4.5: Programs for Empirical Learning.* Morgan Kaufmann, San Francisco, CA, 1993.

[25] Dietrich Wettschereck. *A Study of Distance-Based Machine Learning Algorithms.* PhD thesis, Department of Computer Science, Oregon State University, Corvallis, Oregon, 1994.