Harvesting Classification Trees for Drug Discovery ⋆

Yan Yuan [a], Hugh A. Chipman [b], and William J. Welch [c]

[a] *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*

[b] *Department of Mathematics and Statistics, Acadia University, Wolfville, Nova Scotia B4P 2R6, Canada*

[c] *Department of Statistics, University of British Columbia, 333-6356 Agricultural Road, Vancouver, BC, V6T 1Z2, Canada*

Abstract

Millions of compounds are available as potential drug candidates. High throughput screening (HTS) is widely used in drug discovery to assay compounds for a particular biological activity. A common approach to analysis of the assay data is to build a classification tree to predict the activity of unscreened compounds and hence select further compounds for assay. In many assays, the response variable is a binary indicator of biological activity; the explanatory variables are chemical descriptors capturing compound structure. A tree model will often provide good prediction relative to other methods. It is also relatively interpretable, which is key, since it is of interest to identify diverse chemical classes amongst the active compounds, to serve as leads for drug optimization. Interpretability of a tree is often reduced, however, by the sheer size and number of variables involved. We develop a "tree harvesting" algorithm to reduce the complexity of the tree. Using data from the

National Cancer Institute, we illustrate that an explanatory variable used to build part of a classification tree, may not necessarily be important. Unlike tree pruning, tree harvesting allows such variables to be removed near the top of the tree. The algorithm also aims to reorganize the tree nodes for the interesting "active" class into more coherent groups, thus facilitating identification of the mechanisms for activity.

Key words: Classification tree, high-throughput screening (HTS), sequential screening, tree pruning.

1 Introduction

For drug discovery, the pharmaceutical industry has widely adopted high throughput screening (HTS) to assay millions of compounds for a given target biological activity. Throughput capacities of 100,000 compound/day/assay are common. The purpose of HTS is to identify candidates that can be modified to produce new and effective drugs.

Exhaustive screening of compounds via HTS has many obstacles, however. First, the number of possible drug-like compounds is huge, many orders of magnitude greater than the largest chemical collections. Second, active compounds are very rare. The hit rate, or proportion of active compounds among those screened, is often of the order 1% for a biological target, making the screening process extremely inefficient. Third, valuable resources like proteins

⋆ Research supported by MITACS and NSERC, Canada.

Email addresses: y4yuan at uwaterloo dot ca (Yan Yuan), hugh dot chipman at acadiau dot ca (Hugh A. Chipman), will at stat dot ubc dot ca (William J. Welch).

and chemicals from the inventory are consumed. Fourth, the search is not only for high potencies but also for a variety of chemical structures associated with high potency. Chemists need multiple chemical classes as starting points for modification, because compounds, besides being potent, need to meet constraints on toxicity, side effects, and duration of effect and specificity (Bradley et. al. 2000).

Sequential screening has been advocated by various authors (e.g., Abt et. al. 2001, Engels and Venkatarangan 2001, and Young et. al. 2002) to overcome the obstacles mentioned above and hence improve the efficiency of the drug screening process. In the sequential screening paradigm, the process starts with screening an initial sample (Young et. al. 2002). The objective of the initial screen is to build a statistical model for the structure-activity relationship (SAR), where the explanatory variables are descriptors that characterize physical and chemical properties of a compound and the response variable is its biological activity. If the activity is simply labeled as active/inactive, we have a classification problem. Based on the first statistical model, the biological activities of unscreened compounds are predicted, in order to get a more focused set of compounds with higher probability of activity for a second round of screening. Several cycles of screening-modelling-screening can be used to increase the hit rate of active compounds.

Various statistical models have been investigated for sequential screening, such as classification/regression trees (also known as recursive partitioning), cluster analysis, neural networks, and genetic algorithms (Engels and Venkatarangan 2001). Several successful applications of tree models for SAR modelling have been reported (e.g., Abt et. al. 2001, Young and Hawkins 1998, and van Rhee et. al. 2001), and we concentrate on classification trees in this article.

Classification tree algorithms (Breiman et. al. 1984, Hawkins and Kass 1982) split the space of explanatory variables successively into smaller hyper-rectangles. They are data-adaptive, automatically approximating complex nonlinear relationships, including interaction effects, given enough data. One practical difficulty, however, is deciding on a tree size. Breiman et. al. (1984) tackled this issue by growing a large tree and pruning it back, whereas Hawkins and Kass (1982) used hypothesis testing at each iteration to decide whether to continue partitioning the explanatory variable space. The C4.5 algorithm of Quinlan (1993) has a tree growing stage similar to that of Breiman et. al. (1984).

As mentioned already, in addition to finding active compounds, high throughput screening aims to identify a diverse set of active compounds with different chemical structures and possibly different activity mechanisms. For example, an anti-viral compound could either block a critical enzyme or block the receptors on the host cell membrane that the virus uses to enter the host cell, etc. These underlying mechanisms are totally different for a compound's activity, and the relevant chemical structures could be unrelated. Hence, it is possible that some explanatory variables are important for one mechanism but irrelevant for other mechanisms. Reorganizing a tree so that the important explanatory variables and their critical ranges are more one-to-one with the activity mechanisms is the major thrust of this article. Although such complexities in drug discovery data are the motivation for the research, we anticipate that the potential application areas are much wider.

Like pruning in Breiman et. al. (1984), the proposed "tree harvesting" algorithm simplifies a large tree. Unlike pruning, which works from the bottom of the tree up, however, harvesting allows global reorganization of the tree.

The tree is simplified by removing redundant rules or explanatory variables. Quinlan (1993, Chapter 4) also edited the decision rules of a tree, but the method is quite different. We discuss these distinctions further at the end of Section 2, when our algorithm has been made clearer.

The rest of the article is organized as follows. A simulated example is given in Section 2 to describe the tree method and introduce our tree harvesting algorithm. The algorithm is more formally defined in Section 3. Section 4 compares the results before and after harvesting for the simulated example and for two drug-discovery data sets. The performance of the tree harvesting algorithm is assessed with the latter two data sets. Finally, Section 5 concludes with a discussion of the main findings and future work.

2 Simulated Example

A simple, simulated example (Lam et. al. 2001) will briefly review classification trees and outline our harvesting algorithm.

There are three explanatory variables, LogP, MeltPt, and MolWt, as set out in Table 1. There are two regions of activity, relating to two distinct mechanisms, in the explanatory variable space:

A: $3.5 \leq \text{LogP} \leq 4.0$, and

B: $160 \leq \text{MeltPt} \leq 205$ and $400 \leq \text{MolWt} \leq 500$.

Compounds with explanatory variables inside these two regions are active with a nonzero probability; outside these two regions there are no active compounds.

Table 1

Explanatory variables for the simulated example.

Variable	Description	Range
LogP	Octanol/water partition coefficient	-2 to 7
MeltPt	Melting point	120 to 280 °C
MolWt	Molecular weight	200 to 800

Specifically, the training data for 200 compounds are generated as follows. First, 175 inactive compounds are randomly and uniformly distributed in the three dimensional space of explanatory variables in Table 1. Then, 15 active compounds are generated in the Region A for Mechanism A and another 10 actives are generated in Region B for Mechanism B. These two regions will also contain inactive compounds, since they are uniformly distributed over the entire space. The features of this data resemble those in actual drug discovery problems: several underlying mechanisms, non-linear effects, thresholds, imbalance of classes, and noise. Figure 1 plots the realized training data.

In practice, the various activity mechanisms are not distinguished in the data, where there are just inactive and active compounds. Thus, the data set to be analyzed looks like Figure 2. There is now no obvious pattern.

Uncovering the two active regions in Figure 2 is not straightforward for widely used classification methods. First, we do not know whether multiple activity mechanisms exist, or how many to expect. Second, the set of explanatory variables that are useful versus irrelevant depends on the mechanism. Third, the method needs to adapt to non-linear, threshold effects.

Tree models (e.g., Venables and Ripley 1999, pp. 303–327) are promising for

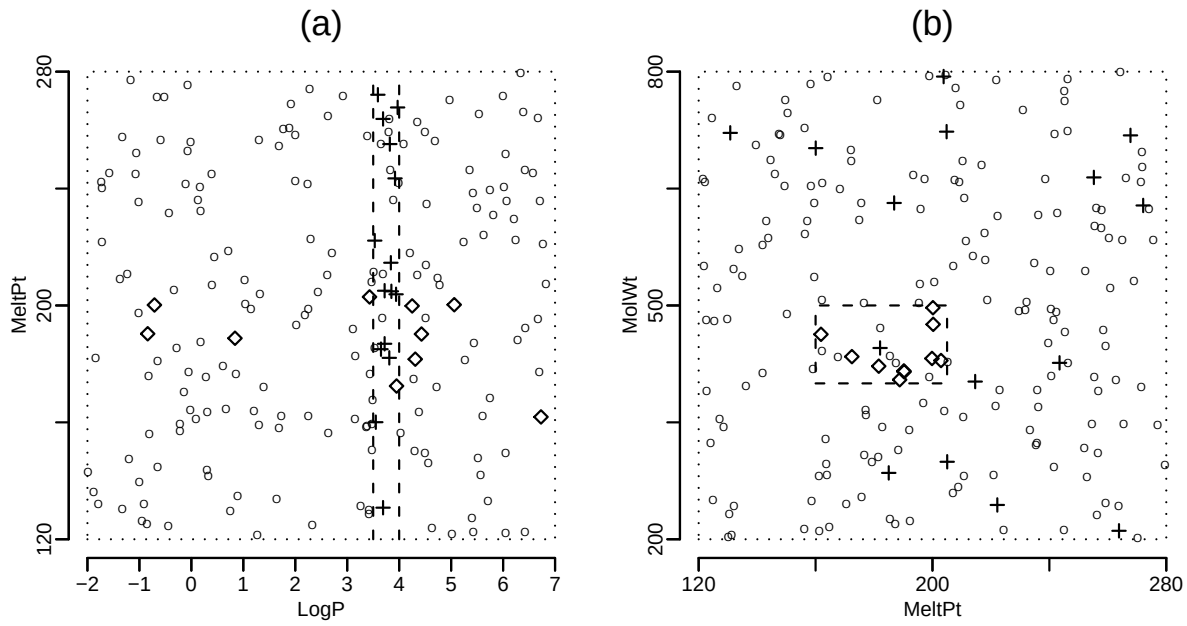

Fig. 1. *Training data for the simulated example. Inactive compounds are shown by circles in the explanatory variable space; compounds active via mechanisms A or B are shown by "+" or ⋄, respectively. In panels (a) and (b) the dashed lines mark the active regions for LogP (Mechanism A) and for MeltPt and MolWt (Mechanism B), respectively.*

drug discovery data with irrelevant variables and threshold effects (Young and Hawkins 1998). A tree model for the simulated example is shown in Figure 3. It is produced by the `rpart` function (Therneau and Atkinson 2006) in R (R core development team, 2006), which allows only binary splits. The ellipse at the top of the tree is the root node, containing all 200 observations, 175 are class 0 (inactive) and 25 are class 1 (active). Because inactives are in the majority, the node is deemed to be class 0. The root node is split into two descendant nodes using the rule LogP < 3.425 versus LogP ≥ 3.425. This rule is chosen from all possible explanatory variables and all possible split boundaries to make the descendants as internally homogeneous as possible. Homogeneity is defined as the change in the log likelihood based on a Bernoulli probability model. It is

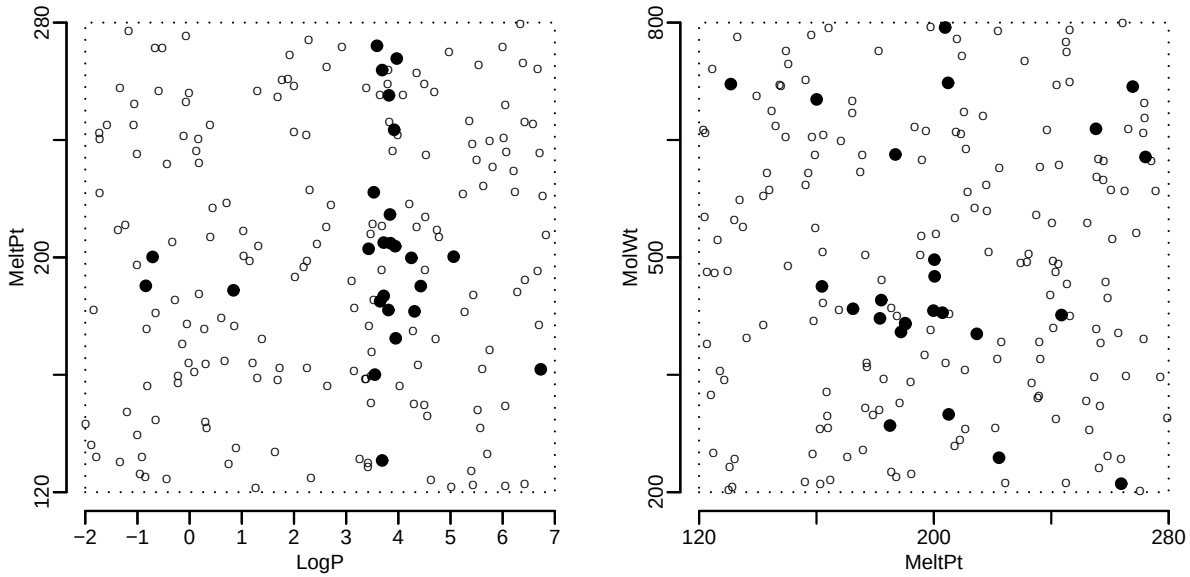

Fig. 2. *Training data for the simulated example. Inactive compounds are shown by small open circles in the explanatory variable space. The two mechanisms for activity are not distinguished; all active compounds are shown by a large solid circle.*

seen that 22 of the 25 active compounds go to the right descendant. Ideally, the two descendant nodes would both be pure—all the inactive-class data going to one node, and all the active-class data going to the other. One split is insufficient, however, and the algorithm iterates, splitting each descendant node. Ellipses denote internal nodes, which are split further by this process. The rectangles represent terminal nodes, where splitting stops.

We now consider how the 12 terminal nodes relate to the two activity mechanisms. Four terminal nodes are classified as active or Class 1. The only active terminal node on the left hand side of the tree, labelled "Node 1", is defined by the rules:

$$\text{LogP} < 3.425 \tag{1}$$
$$188.1 \leq \text{MeltPt} < 218.5 \tag{2}$$
$$400.7 \leq \text{MolWt} < 500.3. \tag{3}$$

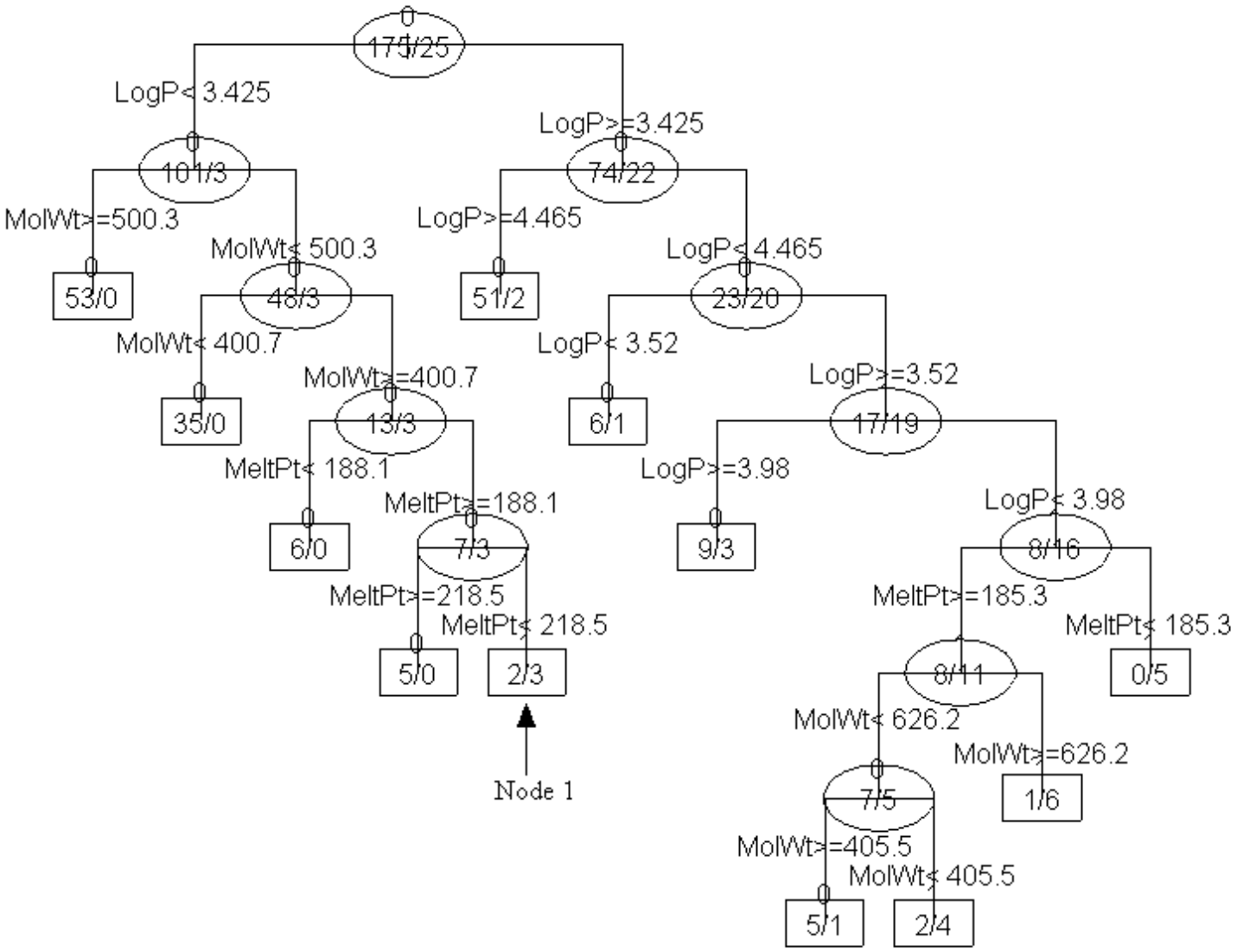

Fig. 3. *Tree for the simulated example. The predicted class for each terminal node is indicated by a 0 or 1 directly above the rectangle. Class 1 nodes are the three rightmost terminal nodes and the node labelled "Node 1".*

It contains a subset of the active compounds from Mechanism B. Similarly, the three active terminal nodes on the right hand side of the tree together have

active compounds coming mainly from Mechanism A, with only one active from Mechanism B. Six active compounds are distributed among the inactive terminal nodes, one from Mechanism A and five from Mechanism B.

We can understand why the tree is unable to identify the two mechanisms by looking at the first split, that of the root node. From Figure 3 we see that the first split is based on LogP and the cutoff is 3.425. This rule corresponds roughly to the left boundary of Region A in Figure 1. While this succeeds in keeping the active compounds for Mechanism A together in the right descendant, the actives from Mechanism B are separated between the two descendants. This root node splitting rule is passed down the tree to all terminal nodes. Thus, it is impossible for the actives from Mechanism B to be grouped together well. Similarly, if either MeltPt or MolWt had been used to split the root node, the Mechanism A actives would have been divided.

The tree harvesting method we propose is a straightforward remedy for this problem. For every terminal node, we consider a "simplification" in which rules corresponding to one or more variables are discarded. For instance, let us examine the active terminal node labelled Node 1 in Figure 3. If rule (1) is removed from the rule set, leaving only rules (2) and (3), a new, larger set of points in the explanatory variable space is defined. These points become Node 1* at the top of Figure 4. The terminal nodes denoted by circles in Figure 4 have fewer data points than before. In addition to all the compounds in Node 1, five further active compounds from four terminal nodes on the right side of the original tree fall in Node 1*. This appears to be an improvement since no inactive compounds move to Node 1*. Moreover, four of the active compounds are removed from inactive terminal nodes. Thus, active compounds, which are of interest here, are gathered together in Node 1*,

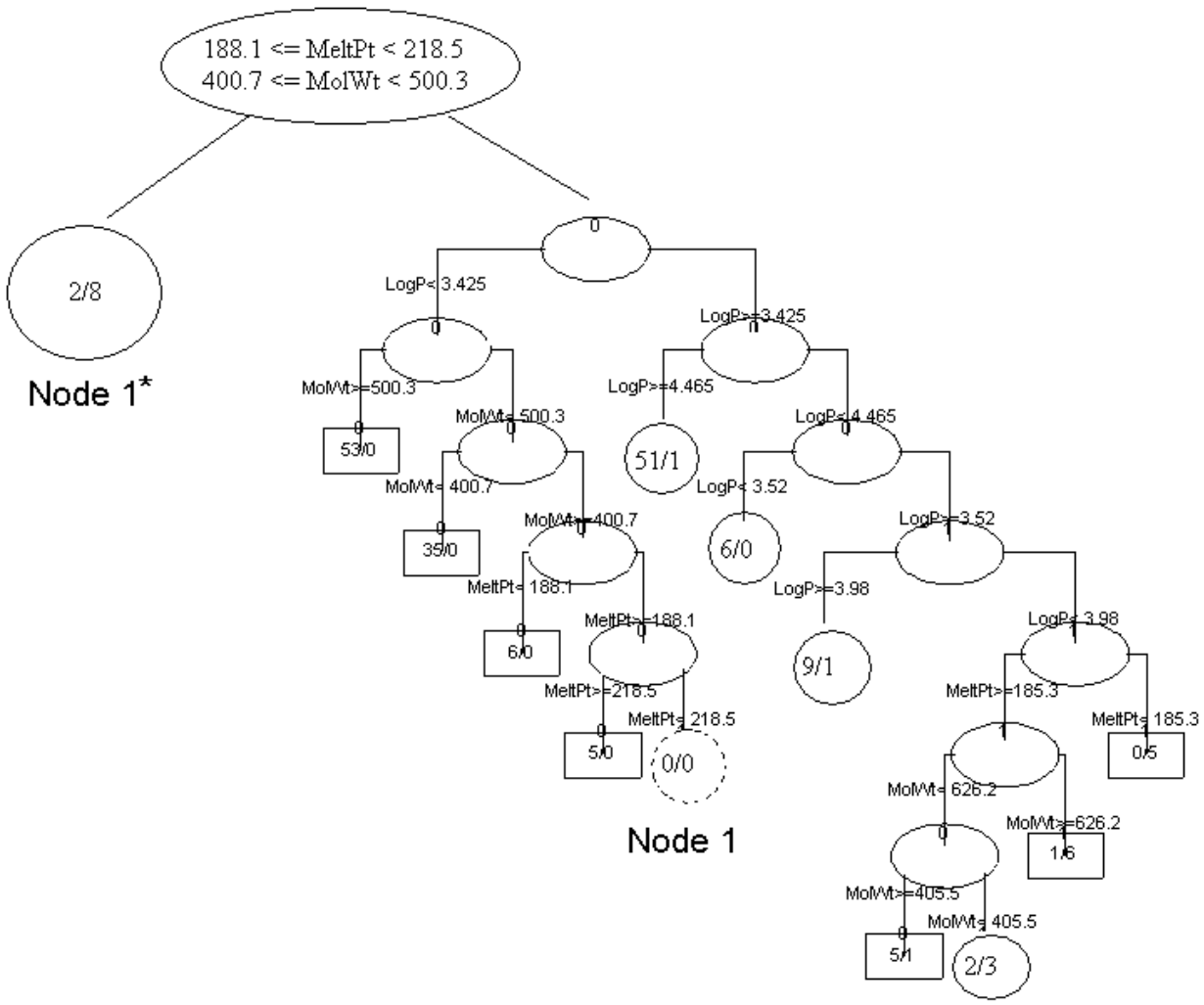

Fig. 4. *The harvested tree after removing rule (1) from Node 1 in Figure 3. Terminal nodes that have changed are indicated by a circle. Node 1 in Figure 3 is now empty.*

hence the term "harvesting".

Alternatively, we could remove one of the other rules in Node 1 or a rule from any other terminal node. To assess which choice is best, we used the log likelihood criterion. For example, when rule (1) is removed from Node 1, five

terminal nodes change. Based on the same Bernoulli probability model used to generate the original tree, the change in log likelihood from Figure 3 to Figure 4 is

$$\begin{aligned}\Delta &= l(2,8) + l(0,0) + l(51,1) + l(6,0) + l(9,1) + l(2,3)] \\ &\quad - [l(2,3) + l(51,2) + l(6,1) + l(9,3) + l(2,4)] \\ &= -16.6 - (-25.3) = 8.7,\end{aligned}$$

where

$$l(a,b) = a\ln(a/(a+b)) + b\ln(b/(a+b)), \tag{4}$$

and $l(0,0) = 0$. The positive value ($\Delta = 8.7$) indicates that this simplification actually yields an improvement in the overall fit of the tree.

After removing this rule from Node 1, we consider whether either of the remaining rules on MeltPt or MolWt can be removed. The changes in log likelihood are $\Delta = -8.92$ and -16.72, respectively (in both cases the resultant tree is inferior). Since both changes fall below $\Delta_{\text{crit}} = -\frac{1}{2}\chi^2_{2,0.95} = -2.99$, the simplification process for Node 1 terminates without deleting either of the remaining two rules. Two degrees of freedom (df) are used for the cutoff since each rule in question has 2 constraints.

If the same calculation is performed for every terminal node in the original tree, it turns out that simplifying Node 1 by removing only rule (1) gives the largest value of Δ.

Harvesting can be applied to the tree iteratively, as shown in Figure 5, until none of the nodes remaining in the original tree can be simplified. We will examine the resulting final harvested tree for the simulated example in Section 4.1.

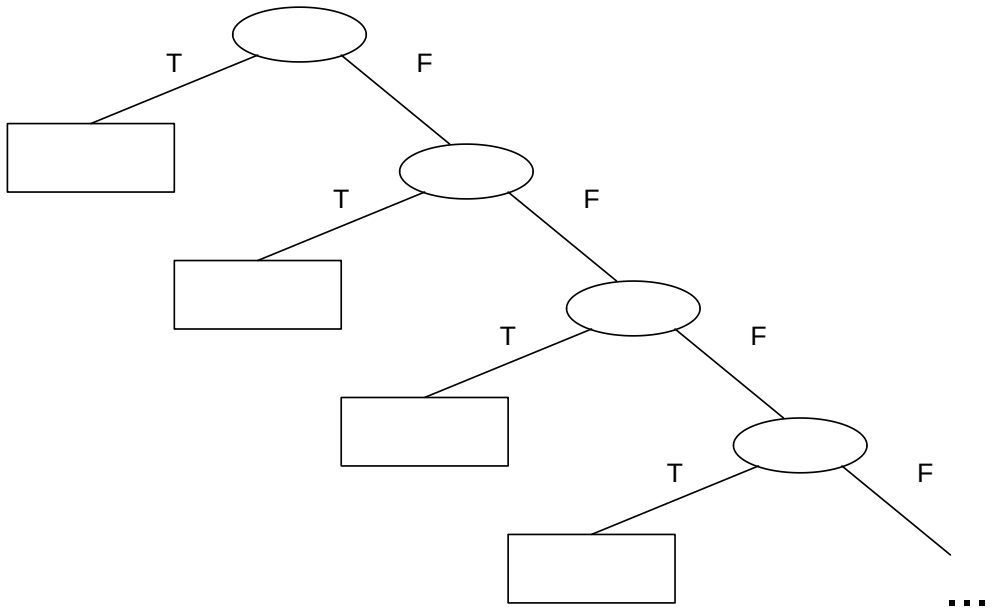

Fig. 5. *A tree after several harvesting iterations. Each ellipse is a rule set that gives the rectangular harvested terminal node as its left descendant.*

In C4.5, Quinlan (1993) also considered simplifying rules for terminal nodes. His method and our method are different in several respects. First, his definition of a rule is any condition that leads to an internal or terminal node. For example, there would be five rules for Node 1 in Figure 3. In our definition, there is at most one rule for a variable (possibly with both lower and upper bounds). Second, and most importantly, Quinlan's decision to delete a rule was based on a criterion similar to the misclassification rate for a single node. In contrast, harvesting looks at the global change in log likelihood for the entire model.

3 Harvesting Algorithm

We now describe the algorithm for the general two-class classification problem with p explanatory variables, but first we need some definitions.

At any stage in the algorithm, let $\mathcal{N} = \{N_1, \ldots, N_t\}$ be a subset of $t = |\mathcal{N}|$ terminal nodes from the original tree; we call these unharvested nodes. Similarly, let $\mathcal{H} = \{\widetilde{N}_1, \ldots, \widetilde{N}_h\}$ be a set of harvested nodes. The tree harvesting algorithm iteratively removes a node from $\mathcal{N}$ and places a simplified version of the node in $\mathcal{H}$.

Node N_i is defined by a set of $k = |N_i|$ rules $R_{i1}, \ldots, R_{ik}$, where $k \leq p$. A rule refers to both the lower and upper constraints on the variable. For example, suppose there are $p = 3$ explanatory variables; a node could be defined by $k = 2$ rules $x_2 > 2.5$ and $0.1 < x_3 < 14.3$. The distinction between $<$ and $\leq$ is unimportant for tree models, because cutoff points are usually chosen mid-way between distinct observed values.

A harvested tree is represented by the set of all nodes, $\{\mathcal{H}, \mathcal{N}\}$. The ordering of the nodes is important. When a point (set of values for the explanatory variables) is passed down the tree, if it satisfies the rules for $\widetilde{N}_1$ then it is assigned to that node. Otherwise, the rules for $\widetilde{N}_2, \widetilde{N}_3, \ldots$ are tested, and the point is assigned to the first node where the rules are satisfied. Only if the point does not belong to any of the harvested nodes is it assigned to one of the unharvested nodes.

The log likelihood for a tree,

$$l(\mathcal{H}, \mathcal{N}) = \sum_{i=1}^{h} l(\widetilde{N}_i) + \sum_{i=1}^{t} l(N_i),$$

is the sum of contributions over all nodes. The contribution, $l(N)$, from any node depends only on a and b, the numbers of Class 0 and Class 1 points, and is given by (4).

A key step in the algorithm is the creation of a new harvested node, $\widetilde{N}_{h+1}$ from one of the unharvested nodes, N_i, by removing one or more of its rules, $R_{i1}, \ldots, R_{ik}$. As the rules for $\widetilde{N}_{h+1}$ are a relaxation of those for N_i, and harvested nodes are considered first in the tree, N_i is depleted of points when harvested, and it no longer makes contribution to the log likelihood and can be deleted from $\mathcal{N}$.

The harvesting process is divided into two algorithms: `harvest` and `simplify`, described in Figure 6. In this figure, set operators $\cup$ and $\setminus$ denote addition/deletion of a node to/from a set of nodes.

The `simplify` function operates on a single node, attempting to remove redundant rules. This is accomplished by deleting the least important rule, one rule at a time, until all remaining rules are significant. The `harvest` function moves one node at a time from $\mathcal{N}$ to $\mathcal{H}$. It identifies the best node N_{i^*} (with simplified form $\widetilde{N}_{i^*}$) to move, and provided that the node has actually been simplified ($\widetilde{N}_{i^*} \neq N_{i^*}$), it will remove N_{i^*} from $\mathcal{N}$ and insert a simplified version $\widetilde{N}_{i^*}$ in $\mathcal{H}$.

The fact that the `simplify` step relaxes the constraints on a node implies that after each harvesting, membership of points in all nodes of $\mathcal{N}$ can change. Thus, every time a node is harvested, the search for the next node to harvest involves trying a simplification of each remaining node in $\mathcal{N}$. Also, every time a rule is removed inside `simplify`, all other rules must be re-evaluated to determine the next best rule to remove.

```
harvest(𝒩) {
    ℋ ← ∅
    do {
        for i = 1 ... |𝒩| {
            Ñ_i ← simplify(N_i; 𝒩, ℋ)
            if (Ñ_{i*} ≠ N_{i*}) Δ_i ← l(𝒩 \ N_i, ℋ ∪ Ñ_i) − l(𝒩, ℋ)
            else Δ_i ← −inf
        }
        if (max(Δ_i) ≠ −inf) {
            i* ← argmax_i Δ_i
            𝒩 ← 𝒩 \ N_{i*}
            ℋ ← ℋ ∪ Ñ_{i*}
        }
    } while (|𝒩| > 0 and max(Δ_i) ≠ −inf)
    return 𝒩, ℋ
}

simplify(N_i; 𝒩, ℋ) {
    Ñ_i ← N_i
    do {
        for j = 1 ... |Ñ_i| {
            N_try ← Ñ_i \ R_j
            Δ_j ← l(𝒩 \ N_i, ℋ ∪ N_try) − l(𝒩 \ N_i, ℋ ∪ Ñ_i)
        }
        j* ← argmax_j Δ_j
        if (Δ_{j*} > Δ_crit) {
            Ñ_i ← Ñ_i \ R_{j*}
        }
    } while (|Ñ_i| > 0 and Δ_{j*} > Δ_crit)
    return Ñ_i
}
```

Fig. 6. *Pseudo code for the harvested algorithm. Details and notation are provided in Section 3.*

In `simplify`, the critical value Δ_{crit} will correspond to $-\frac{1}{2}\chi^2_{m,0.95}$, half of a 95% upper quantile of a Chi-square distribution with m df, with $m = 1$ or 2 depending on whether the rule being deleted is 1-sided or 2-sided.

Table 2
Harvested nodes for the simulated example.

Harvested	Variable range			
node	LogP	MeltPt	MolWt	Inactive/Active
1	[- , -]	[188.1, 218.5]	[400.7, 500.3]	2/8
2	[- , 3.425]	[- , -]	[- , -]	99/0
3	[3.52, 3.98]	[- , -]	[626.2, -]	1/8
4	[3.52, 3.98]	[- , -]	[- , 405.5]	2/4
5	[- , -]	[185.3, -]	[- , -]	46/1
6	[3.52, 3.98]	[- , -]	[- , -]	0/2
7	[- , -]	[- , -]	[- , -]	25/2

4 Results

4.1 Simulated example

Applying the harvesting algorithm to the simulated data in Section 2 leads to the harvested tree in Table 2. The choice of harvested node 1, labelled node 1* in Figure 4, was described in Section 2. It contains two inactive compounds and eight active compounds in the training data. Harvested node 1 is roughly a subset of the actives from Mechanism B.

Harvested node 2 concentrates 99 of the inactive compounds together, with no actives. Its rules define essentially the inactive area to the left of the LogP active region in Figure 1(a). Note that there are three active Mechanism B compounds in this area of Figure 1(a), but they are picked up by harvested node 1.

The rules defining harvested nodes 3, 4, and 6 all contain exactly the LogP rule defining Mechanism A. Two of these nodes also include rules to exclude Mechanism B active compounds.

Harvested nodes 5 and 7 are inactive nodes, with three active compounds between them. In contrast, six active compounds were hidden in active nodes in the original tree of Figure 3.

After harvesting seven terminal nodes, all 12 terminal nodes of the original tree are depleted.

The harvested tree is therefore largely successful in separating the two mechanisms. Mechanism A corresponds to harvested nodes 3, 4, and 6, while Mechanism B corresponds approximately to harvested node 1. The harvested tree is also more successful in classifying active compounds overall: only three of the 25 active compounds fall in inactive nodes.

4.2 NCI and Mutagenicity Data Sets

We now illustrate the algorithm with two binary classification problems. The NCI data has six explanatory variables and extremely unbalanced classes. The mutagenicity data has 47 explanatory variables and balanced classes.

4.2.1 Data descriptions

The NCI data set consists of 29,812 compounds that were measured for their ability to protect human CEM cells from HIV-1 infection. The database is available from the National Cancer Institute, at `http://dtp.nci.nih.gov/docs/aids/aids_data.html`. The 1999 release of data is used in our analysis. The original response contained three categories: Inactive (29,204 compounds), moderately active (393 compounds) and active (215 compounds). The two active classes were merged to form a single "active" class, somewhat mitigating the problem of class imbalance. The BCUT descriptor set (Burden

1989, Pearlman and Smith 1999) is calculated (by GlaxoSmithKline chemists) to serve as explanatory variables. BCUT metrics are an extension of Burden's parameters which are based on a combination of the atomic number for each atom and a description of the nominal bond-type for adjacent and nonadjacent atoms (Burden 1989, Pearlman and Smith 1999). The six BCUTs used in this study describe the compounds' atomic charge, polarizability, and hydrogen bond donor-acceptor ability (Todeschini and Consonni 2000). This set of BCUT numbers has been successfully used in modelling the NCI AIDS data (Lam et. al. 2001, Wang et. al. 2002).

In the mutagenicity data, the binary response variable codes mutagenic/non-mutagenic. Since mutagenic compounds tend to increase the frequency or extent of DNA mutation, it is cost effective to identify mutagens in the early phase of drug development process. Here, we analyze a dataset of 1866 compounds (968 non-mutagenic, 898 mutagenic) obtained from GlaxoSmithKline. The constitutional descriptor set used for the explanatory variables is generated using Dragon software (Todeschini and Consonni 2000). The constitutional descriptor set consists of 47 variables such as molecular weight, volume, number of oxygen atoms, etc.

4.2.2 Analysis of NCI data

The `rpart` library (Therneau and Atkinson, 2006) in R was used to build the tree model. Previous work on the NCI data (Wang et. al. 2002) found that the best out-of-sample predictive performance was achieved by growing very large trees without any pruning. Only node size was restricted (5 or more observations per terminal node, and at least 10 observations are required to split a node).

The original tree has 124 terminal nodes, while the harvested tree has 122 terminal nodes. Although harvesting does not dramatically reduce the number of nodes, it does simplify the rules making up each terminal node: the original tree had an average of 4.54 rules/node, while the harvested tree has 2.5 rules/node. In these counts, a "rule" is defined as 1 or more conditions on an explanatory variable, thus limiting the number of rules above by the number of explanatory variables (6 for NCI). Although the number of nodes is large, only a few are likely to be examined closely, since only a small fraction of the total nodes have high activity levels. Short rules will aid interpretation of these nodes.

As the nodes are harvested, the change in log-likelihood can be monitored. Figure 7 shows that the changes are small, positive quantities, implying that even though rules are being simplified, small *improvements* in fit are being gained. The overall log likelihood change is 48.6, indicating a substantial improvement from the original tree model. A formal test of significance is not carried out, because of difficulty in identification of an appropriate reference distribution.

To accurately assess predictive performance, out of sample (i.e., test set) accuracy must be evaluated. The modelling described thus far was in fact carried out on a randomly selected 50% subset of the full dataset. This division of the data into train/test sets was constrained so that each set had the same balance of active/inactive compounds. We now consider predictive performance on the test set. Before giving performance results, we briefly discuss appropriate techniques for assessing predictive performance in unbalanced classification.

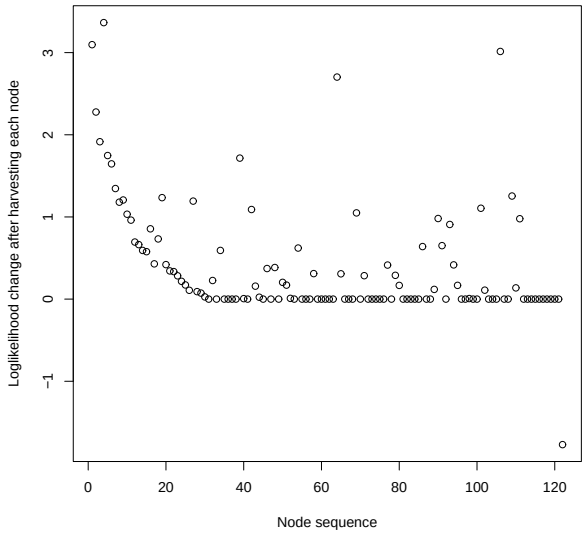

Fig. 7. *The sequence of log likelihood change as node is harvested from the first to the last.*

Measuring performance in unbalanced classification

We will consider a performance measure specifically designed to assess the ability of the model to accurately identify active compounds. We note in passing that misclassification rate is inappropriate for unbalanced problems: a very low misclassification rate can be obtained by a naive classifier that always predicts the majority class.

We describe briefly the hit curve, a graphical measure of performance (see also Zhu et. al. 2006). In its construction, compounds are first ranked according to their predicted probability of being active. These compounds are then "selected", one at a time, and the cumulative number of actives recorded. A hit curve displays the number selected on the horizontal axis against the number of actives on the vertical axis. Steeply rising hit curves indicate a model effective at selecting active compounds, while a line with slope $\frac{n_1}{n}$, where n_1 denote the number of actives and n denotes the number of total compounds,

corresponds to selecting compounds in random order. Tied predictions are typically handled by selection of groups of tied points rather than one-at-a-time selection. When tied groups arise, the ranking problem becomes one of ranking groups rather than individual observations.

Two ranking criteria are used producing two hit curves in this study. One is $\hat{p}$, the estimated probability of being active. For both trees and harvested trees, $\hat{p}$ for each terminal node is the sample proportion of actives in that node. Predictions are then made by passing each new compound down the tree, and assigning the $\hat{p}$ from the node in which the compound lands. The main problem with ranking by $\hat{p}$ is that estimation uncertainty is ignored. Consider two terminal nodes, one having 100 compounds with 99 active ($\hat{p}$= 99/100 = 0.99), the other having only one compound which is active ($\hat{p} = 1/1 = 1.00$). The $\hat{p}$ criterion will give the second node higher rank, even though its $\hat{p}$ is more variable. Lam et. al. (2001) encountered this problem in a similar context, and ranked nodes using a lower confidence bound on the probability of activity. Assuming a Binomial model for responses in each terminal node, we shall use an exact 95% lower confidence bound, p_{lb}. In a node with m actives out of n compounds, p_{lb} can be calculated as the solution for p in the equation $\Pr(M \geq m|n, p) = 0.05$, assuming that $M \sim \text{Bin}(n, p)$. Under this criterion, the p_{lb} is 0.95 for the 1/99 node and 0.05 for the 0/1 node.

We now return to a comparison of predictive accuracy. Figure 8 displays test-set hit curves for the original tree and for the harvested tree. Two different curves are displayed for each model, corresponding to ranking by $\hat{p}$ and by p_{lb}. When we select more than 250 compounds, the harvested tree dominates the original tree regardless of criterion used. For a smaller number of compounds selected, the original tree with either ranking criterion is comparable to the

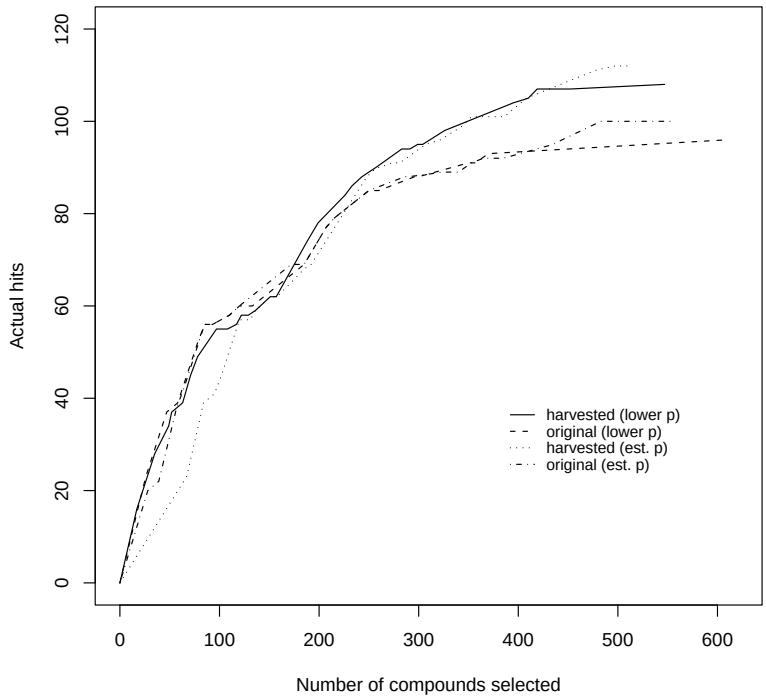

Fig. 8. *Actual number of hits as compounds are selected from the test data by the tree and harvested tree model (NCI data).*

harvested tree with p_{lb} ranking. Overall, the harvested tree with p_{lb} ranking is very close to optimal no matter how many compounds are selected, and beats the original tree when more than 250 compounds are selected.

4.2.3 Analysis of Mutagenicity Data

The balanced nature of the mutagenicity data leads to some different techniques for model-building and performance assessment. The `rpart` tree was pruned using cross-validated misclassification rate, and the harvesting algorithm applied to the pruned tree. Due to the balanced nature of the data, predictive performance will be measured with misclassification rate.

Table 3 shows some characteristics of the original tree and harvested tree. Five different partitions of the data into training and test sets are reported. In the

Table 3
Results of tree algorithm and harvesting algorithm for five different train/test splits of the mutagenicity data set. In the first row all data was used for model-building, while in the other rows 25% was used for testing.

Train/	# of Nodes		Rules/node		Test set misclass rate		# of variables used	
Test	H. Tree	Tree	H. Tree	Tree	H. Tree	Tree	H. Tree	Tree
abcd/	20	22	2.25	4.73	n/a	n/a	15	15
bcd/a	18	19	2.11	5.42	0.275	0.290	13	13
acd/b	19	22	2.42	5.27	0.236	0.240	16	16
abd/c	24	25	2.71	5.28	0.262	0.266	14	14
abc/d	24	25	2.25	6.08	0.218	0.212	18	18

first instance, all data were used for training. In the other instances, a 75/25% train/test split was used. Within the training set, 3-fold cross-validation (each time leaving out 25% of the full data) was used to select tree size. In all instances, the same four subsets (labelled a-d) are used as building blocks for the train/test splits, and for cross-validation within the training set. In all cases, the harvested tree has (i) slightly fewer nodes, (ii) half the number of rules in each node, (iii) comparable or better test set misclassification rates, and (iv) no reduction in the number of variables used. These results are similar to the NCI example, with harvesting yielding shorter, more interpretable rules, and a slight improvement in out-of-sample predictions.

The tree constructed using all the data has 22 terminal nodes and uses 15 of the variables. Table 4 shows the first 9 harvested nodes, and lists the variables removed during harvesting. The table indicates considerable simplification of the original tree.

In the original tree, node 1 is constrained by five conditions:

$$2.5 < \text{nAB (number of aromatic bonds)} < 14.5 \tag{5}$$

Table 4
The first 9 harvested nodes for the mutagenicity data.

Node	Mutagenic/ total	Label	Rules	Deleted variables
1	40/65	1	2.5<nAB<14.5; nBM<6.5; nN=1	AMW nR03
2	71/92	1	nAB>14.5; nN>1.5	nCIR
3	53/64	1	nAB<14.5; nR03>0.5	AMW nN
4	15/126	0	nR06>0.5; nAT<15.5	nAB nN nO Sp
5	20/60	0	2.5<nAB<14.5; Mv<0.635; nN=1	AMW nBM nR03
6	96/125	1	nAB<14.5; AMW>6.245; 0.5<nX<3.5	nN nR03
7	9/47	0	nBnz=0; nO=0; nN>1.5	nAB Sp
8	93/132	1	nN>1.5; nBnz>0.5; Sp<20.33	nR03 nX
9	35/131	0	nAB<14.5; nN<1.5; AMW>6.245; MW<221.3	nR03 nX

$$\text{nN (number of Nitrogen atoms)} = 1 \tag{6}$$
$$\text{nBM (number of multiple bonds)} < 6.5 \tag{7}$$
$$\text{nR03 (number of 3-membered rings)} = 0 \tag{8}$$
$$\text{AMW (average molecular weight)} < 6.245. \tag{9}$$

Node 1 contains 49 compounds and 32 of them are mutagenic. When harvested, conditions (8) and (9) on nR03 and AMW were dropped. The resulting harvested node contains 65 compounds, of which 40 are mutagenic. Since there are three rules for this harvested node, the interpretation becomes simpler. Similar complexity reduction occurs in the harvesting of node 2, while increasing the size of the node from 45 to 92 compounds and reducing the node's misclassification rate from 27% to 23%.

Similar to the NCI analysis, the harvesting process yields a sequence of mostly small increases to the log-likelihood. The improvement is largest for the first six nodes, then near zero for the remaining nodes. The overall log likelihood increase is 21.62, indicating a substantial improvement from the original tree

model. These results are for harvesting a tree grown to the full dataset, although the patterns are similar for the four train/test splits as the above.

5 Discussion

The tree harvesting algorithm is an innovative approach to deal with multiple mechanisms. The current algorithm creates simpler rules in harvested nodes, can increase the size of nodes (improving the support of the rules generated), and often demonstrates modest improvements in prediction performance. We have focussed on classification trees, but the same methodology would apply to regression trees for a continuous response variable.

The current harvesting algorithm treats the original rules coarsely, considering the deletion of all rules involving a variable at once. Finer control might be possible, such as adjusting the individual constraints on variables rather than just deleting them. For example, in the simulated example, the true constraint on MeltPt for mechanism B is [160, 205], while the tree identifies the constraint as [188.1, 218.5]. By adjusting (and possibly deleting) the individual bounds in this rule, it may be possible to better uncover the actual structure, and perhaps reduce the number of terminal nodes substantially.

In unbalanced problems, where the goal is to identify a target class, tree harvesting can be valuable because (1) it produces simpler rules and (2) the harvested nodes are often applicable to more data, since rules have been removed. Thus, the first few harvested nodes that are active may be all that need be examined in order to identify sufficient observations from the rare class. Interpretability might be further improved if the harvesting algorithm could be biased to select more "active" nodes early in the harvesting process.

One possible approach for this is to differentially weight classes during the harvesting stage.

The criterion to harvest nodes is based on the log-likelihood, and harvesting continues provided that the change in log-likelihood is larger than a negative cutoff. Although an appropriate reference distribution has not been identified, the current cutoffs are based on the 95th percentiles of a χ^2 distribution with 1 or 2 degrees of freedom, depending on whether 1 or 2 constraints make up the rule being deleted. While identification of a proper reference distribution for this cutoff would be desirable, it is complicated by the greedy nature of the harvesting algorithm.

A number of other approaches have some similarity to our tree harvesting algorithm. Quinlan's C4.5 algorithm has an optional step, c4.5rules (Quinlan 1993, Ch. 5), which allows the deletion of rules from the rulesets that identify terminal nodes. This approach is similar, although our method uses likelihood-based criteria for pruning, considers deletion of entire variables, and includes specialized tools for the analysis of unbalanced classification problems.

Friedman and Fisher's (1999) PRIM algorithm seeks to identify rules of the same form as our tree harvesting algorithm. A critical difference is that PRIM is a top-down algorithm that constructs the final nodes "from scratch", while our approach modifies an existing tree. Both methods have merits and limitations. One advantage of our approach is that it can be used as a complement to a tree-based analysis, producing rules that are more interpretable than the original tree. More recent work by Friedman and Popescu (2005) also considers the generation of rule ensembles, but allows different rules to additively combine to generate a final prediction.

Building models for HTS data on low dimensional subspaces (subsets of variables) was investigated by Lam et. al. (2001) and Wang et. al. (2003). These approaches are based on the hope that only a few variables are relevant to structurally different chemical classes at a time. Their results suggested it is likely the case. The results of this study also support this hypothesis.

Lam et. al. (2001) proposed a cell-based analysis method for HTS data. With their method, preliminary identification of a good cell is achieved by three steps. In step 1, the descriptor (explanatory variable) space is divided into 1D/2D/3D cells. In step 2, cells are shifted to optimize their boundary with respect to active compounds. The shifting of cells provides an effective means of handling different shapes and sizes of active regions. In step 3, the preliminary good cells are identified by counting the number of active compounds in each cell. Our harvested active nodes may be used as initial cells in step 1, since the harvested nodes are often associated with 1-3 variables defining a low dimensional subspace. The harvested active nodes could also be used to guide the dividing process in step 1 by pointing out relevant subspaces. Lam et. al. (2001) considered all possible 1D/2D/3D subspaces. With p descriptors, there are $\binom{p}{1} + \binom{p}{2} + \binom{p}{3} = (5p + p^3)/6$ subspaces in total. For large p, it is suggested that reducing the number of subspaces by focusing on only 1D and 2D subspaces for computational reasons. After we obtained harvested nodes, we can focus on the subspaces defined by them. This could lead to a significant reduction in the number of subspaces to be searched.

References

Abt, M., Lim, Y., Sacks, J., Xie, M. and Young, S. S. (2001), "A Sequential Approach for Identifying Lead Compounds in Large Chemical Databases." *Statistical Science,* 16, 154–168.

Bradley, E. K., Beroza, P., Penzotti, J .E., Grootenhuis, P. D. J., Spellmeyer, D. C., and Miller, J. L. (2000), "A Rapid Computational Method for Lead Evolution: Description and Application to α_1-Adrenergic Antagonists." *Journal of Medicinal Chemistry,* 43, 2770–2774.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees.* Wadsworth, Monterey, CA.

Burden, F. R. (1989), "Molecular Identification Number for Substructure Searches." *Journal of Chemical Information and Computer Science,* 29, 225–227.

Engels, M. F. M. and Venkatarangan, P. (2001), "Smart Screening: Approaches to Efficient HTS." *Current Opinion in Drug Discovery and Development,* 4, 275–283.

Friedman, J. H. and Fisher, N. I. (1999) "Bump Hunting in High-Dimensional Data", *Statistics and Computing*, 9, 123–143.

Friedman, J. H. and Popescu, B. E. (2005) "Predictive learning via Rule Ensembles", Technical report, Stanford University Department of Statistics.

Hawkins, D. M. and Kass, G. V. (1982), "Automatic Interaction Detection." *Topics in Applied Multivariate Analysis,* Hawkins, D. M., ed. Cambridge University Press, Cambridge.

Lam, R. L. H., Welch, W. J., and Young, S. S. (2001), "Cell-Based Analysis of High Throughput Screening Data for Drug Discovery." Research Report

RR-02-02, Business and Industrial Statistics Research Group, University of Waterloo.

Pearlman, R. S. and Smith, K. M. (1999), "Metric Validation And The Receptor-Related Subspace Concept." *Journal of Chemical Information and Computer Science,* 39, 28–35.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California.

R Development Core Team (2006), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, `http://www.R-project.org`.

Therneau, T. M. and Atkinson, B. (2006) "Rpart: Recursive Partitioning", *R package version 3.1-22*, S-PLUS original at `http://www.mayo.edu/hsr/Sfunc.html`.

Todeschini, R. and Consonni, V. (2000), *Handbook of Molecular Descriptors.* Wiley-VCH, Weinheim, Germany.

van Rhee, A. M., Stocker, J., Printzenhoff, D., Creech, C., Wagoner, P. K. and Spear, K. L. (2001), "Retrospective Analysis of an Experimental High-Throughput Screening Data Set by Recursive Partitioning." *Journal of Combinatorial Chemistry,* 3, 267–277.

Venables, W. N. and Ripley, B. D. (1999), *Modern Applied Statistics with S-Plus (3rd edition).* Springer.

Wang, M., Chipman, H., and Welch, W. J. (2002), "Mining Nuggets of Activity in High Dimensional Space from High Throughput Screening Data." Research Report RR-02-01, Institute for Improvement in Quality and Productivity, University of Waterloo.

Wang, M., Chipman, H., and Welch, W. J. (2003), "Classification for Ranking in Drug Discovery: Identifying and Aggregating Relevant Subsets of Vari-

ables." Proceedings of the ISI Conference on Environmental Statistics and Health, Santiago de Compostela, Spain, pp. 173–181.

Young, S. S. and Hawkins, D. M. (1998), "Using Recursive Partitioning to Analyze a Large SAR Data Set." *SAR and QSAR in Environmental Research,* 8, 183–193.

Young, S. S., Lam, R. L. and Welch, W. J. (2002), "Initial Compound Selection for Sequential Screening." *Current Opinion in Drug Discovery and Development,* 5, 422–427.

Zhu, M., Su, W., and Chipman, H. A. (2006) "LAGO: A Computationally Efficient Approach for Statistical Detection", to appear in *Technometrics.*